A comparative review on mobile robot path planning : classical or meta-heuristic methods?

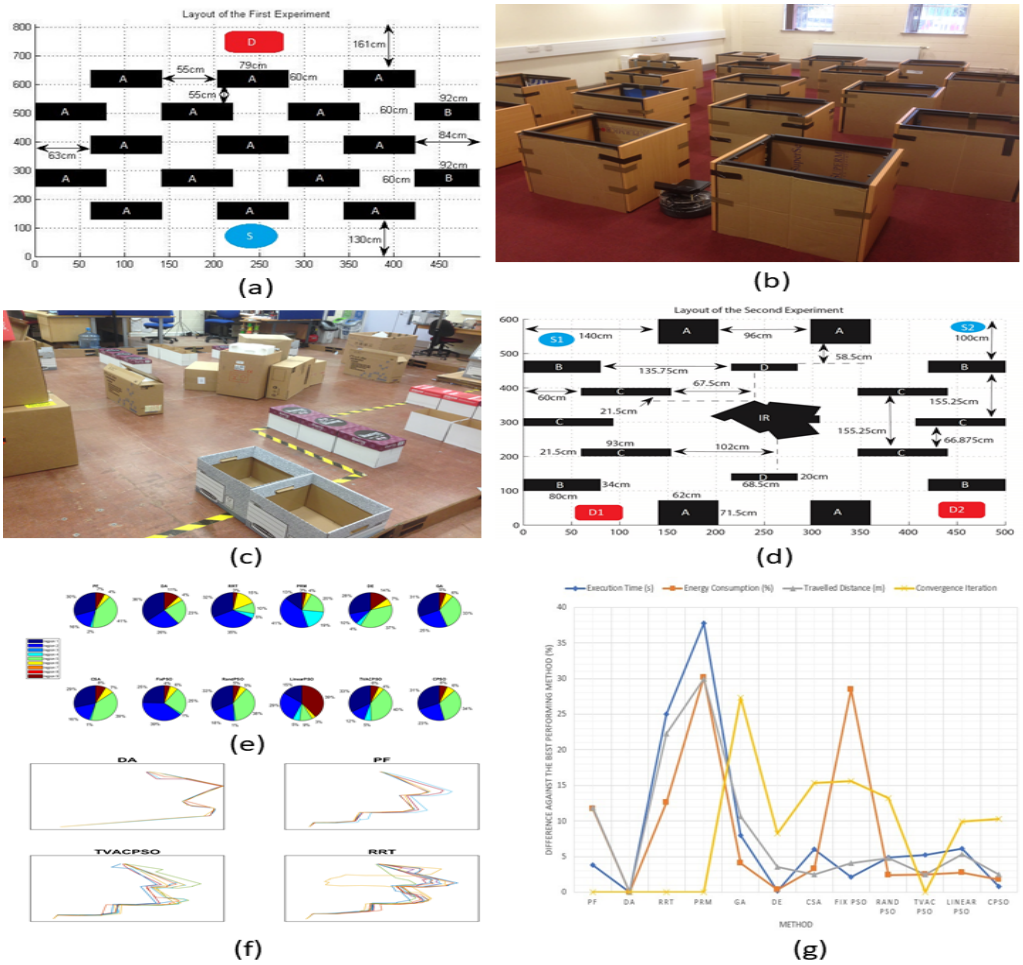
Wahab, MNA, Nefti-Meziani, S and Atyabi, A

http://dx.doi.org/10.1016/j.arcontrol.2020.10.001

Graphical Abstract

A Comparative Review on Mobile Robot Path Planning: Classical or Meta-heuristic Methods?

Mohd Nadhir Ab Wahab, Samia Nefti-Meziani, Adham Atyabi



(a) Experiment 1 Design in Maze Layout
(b) Real Setup of Experiment 1
(c) Experiment 2 Design in Symmetric Layout
(d) Real Setup of Experiment 2
(e) Regions Occupational Percentages based on Robot's Location
(f) Selected Algorithms Local Path Tracking for Experiment
(g) Results Comparison between Algorithms Deployed on Several Factors Considered.

Highlights

A Comparative Review on Mobile Robot Path Planning: Classical or Meta-heuristic Methods?

Mohd Nadhir Ab Wahab, Samia Nefti-Meziani, Adham Atyabi

- Classical and heuristic robot motion planning are assessed
- Complex, obstacle rich and realistic environments are used
- Multiple criterions are considered for performance assessment
- Constrained particle swarm optimization showed superior results in most criterions

A Comparative Review on Mobile Robot Path Planning: Classical or Meta-heuristic Methods?

Mohd Nadhir Ab Wahab[a,*], Samia Nefti-Meziani[b], Adham Atyabi[c]

[a]*School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia*
[b]*School of Science, Engineering and Environment, University of Salford, Greater Manchester, United Kingdom*
[c]*Department of Computer Science, University of Colorado Colorado Springs, United States*

Abstract

The involvement of Meta-heuristic algorithms in robot motion planning has attracted the attention of researchers in the robotics community due to the simplicity of the approaches and their effectiveness in the coordination of the agents. This study explores the implementation of many meta-heuristic algorithms, e.g. Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Cuckoo Search Algorithm (CSA) in multiple motion planning scenarios. The study provides comparison between multiple meta-heuristic approaches against a set of well-known conventional motion planning and navigation techniques such as Dijkstra's Algorithm (DA), Probabilistic Road Map (PRM), Rapidly Random Tree (RRT) and Potential Field (PF). Two experimental environments with difficult to manipulate layouts are used to examine the feasibility of the methods listed. several performance measures such as total travel time, number of collisions, travel distances, energy consumption and displacement errors are considered for assessing feasibility of the motion planning algorithms considered in the study. The results show the competitiveness of meta-heuristic approaches against conventional methods. Dijkstra 's Algorithm (DA) is considered a benchmark solution and Constricted Particle Swarm Optimization (CPSO) is found performing better than other meta-heuristic approaches in unknown environments.

*Corresponding author

Email addresses: mohdnadhir@usm.my (Mohd Nadhir Ab Wahab), s.nefti-meziani@salford.ac.uk (Samia Nefti-Meziani), aatyabi@uccs.edu (Adham Atyabi)

Keywords: Path Planning, Classical, Meta-heuristic, Mobile Robot, Navigation

1. Introduction

Navigation, an important factor in mobile robotics, is defined as the process of identifying the robot's position accurately, planning the path, and following the path planned (Pennock, 2005). Localization is the ability of the robot to determine its exact location in the real-world with respect to its position inside a map; path planning is considered as the computation of a path through a map which represents the environment. This given path is chosen based on the problem objectives so that the expected destination can be achieved. As such, a reliable map is essential for navigation without which robots are not capable of accomplishing their goals. In navigational approaches, the reliability of the map is challenged due to the dynamic and unpredictable nature of the real-world applications (Atyabi et al., 2010; Freund and Kaever, 2017).

This study explores the feasibility of meta-heuristic based approaches for motion planning and navigation of a mobile robot in static environments with numerous obstacles resulting in a hard to manoeuvre path. The main hypothesis is that a mobile robot controlled by a meta-heuristic approach (e.g. Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), and Cuckoo Search Algorithm (CSA)) is expected to perform as well as, or better than, conventional motion planning and navigation techniques (e.g., Dijkstra's Algorithm (DA), Probabilistic Road Map (PRM), Rapidly Random Tree (RRT) and Potential Field (PF)). In order to evaluate the feasibility of the utilised meta-heuristic based methods, two sets of experiments with different layouts are considered. In the first layout, a mobile robot needs to manoeuvre in maze layout setting. In the second experiment, a symmetrical layout with an irregular shape obstacle placed in the middle of the environment is considered. Multiple start and destination locations are considered in this layout resembling paths with varying complexities raised from hard to manoeuvre corridors to obstacle dense environments. The experimental layouts considered in this study are designed to comprehensively attest the feasibility of meta-heuristic based and conventional motion planning methods. The following section presents the problem statement of the study.

Problem Statement. The problem statement for all experiments involved is defined around the navigation of a mobile robot in an unknown environment using a classical or meta-heuristic navigation method. The problem is defined in the following terms:

- Let $A_i(x, v)$ be a robot at location x with velocity v in environment E. i is the agent's solution dimension in the search space. That is, A represent a swarm and each member of the swarm represent a possible solution (robot's next visiting location) in the search space represented by a dimension.
- Robot moves in *Euclidean space*, E called environment represented as R^N, with $N = 3$ (see Fig. 1 and 3).
- Let $O_1, ..., O_n$ be fixed placed obstacles distributed in E with $n = 11$.
- Let S and G be the starting and goal locations in E.

Problem: Given the initial location of A being S and obstacle O_i ($i = 1, ..., n$) being placed on fixed locations in E, find a path from location S to G while avoiding collision with obstacles with the assumption that the robot has no prior knowledge about the location of the obstacles in the environment.

The outline of this study is as follows: the background of the approaches used in this study and their pseudo-codes are discussed in Section 2. Section 3 elaborates on the experimental design. Results and discussion are presented in Section 4. Conclusion and future works are discussed in Section 5.

2. Robot Motion Planning: Classical vs. Meta-Heuristic based methods

Heuristic-based and classical methods are two of the common categories of navigation and motion planning approaches. In classical category, methods such as Cell Decomposition (CD), Potential Field (PF), Road Map and Sub-goal Network are commonly used in motion planning problems (Atyabi and Powers, 2013). These methods are found to be less capable in handling unknown, partially known, or dynamic environments in their basic formulation and are known to be computationally intensive. In addition, they

are mostly found to be dependent on complete prior knowledge of the environment in order to create a feasible path between the starting and the destination points.

Meta-heuristic approaches can overcome these issues due to their proficiency in handling unknown or partially known environments. These methods create a set of temporary paths within each of their iterations, bringing them closer to the destination location one step of the algorithm at a time. The Meta-heuristic methods use different approaches compared to the classical methods where rather than creating a global path based on the prior knowledge (e.g. environment map) and execute it, they create a subpopulation of possible maneuvers in each iteration. Later, the best path created will be chosen based on their fitness and the path will be executed. Atyabi and Powers (Atyabi and Powers, 2013) considered approaches such as Genetic Algorithm (GA), Neural Networks (NN), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) as meta-heuristic based methods for motion planning. Duguleana and Mogan (Duguleana and Mogan, 2016) presented the combination of Neural Networks and Reinforcement Learning to enhance the capabilities of mobile robots to deal with static and dynamic obstacles.

The classical algorithms for motion planning considered in this study are Potential Field (PF), Dijkstra's Algorithm (DA), Rapidly-explore Random Tree (RRT) and Probabilistic Road Map (PRM). These four approaches are chosen because of their well known reputation in the community and being widely and successfully applied in path planning problems (Zafar and Mohanta, 2018; Elbanhawi and Simic, 2014; Zammit and Van Kampen, 2018; Mohanan and Salgoankar, 2018). The overall performances of these methods are known to be modest in global motion planning problems. PF, RRT and PRM are also utilised in local path planning problems (Chen, 2014; Contreras-Cruz et al., 2015; Mohanta and Keshari, 2019). In this study, GA, DE, variants of PSOs and CSA are chosen as candidates in meta-heuristic category due to their popularity and their high potential in addressing local motion planning problems (Atyabi et al., 2010; Zafar and Mohanta, 2018; Patle et al., 2019; Sathiya and Chinnadurai, 2019; Lamini et al., 2018).

One of recent enhanced classical methods for path planning is introduced by Bayat et al. (Bayat et al., 2018), where Electrostatic potential field approach is utilised for mobile robots' path planning. This approach requires complete layout/map of the environment in order to generate a feasible path between the initial location and the final location which is the main draw-

back for classical methods. Nazarahari et al. proposed a hybrid method containing classical (PF) and meta-heuristic (Enhanced-GA) methods. In their approach, PF initially will generate feasible paths, and later EGA will select the best optimal path between starting point and the destination point before the motion planner is executed (Nazarahari et al., 2019). Kamil et al. suggested to equip the robot with proper sensor such as range-finder to deal with not only static obstacles but also dynamic obstacles (Kamil et al., 2017). Mac et al. also discussed about several classical algorithms and heuristic-based algorithms related to path planning, however, all the algorithms discussed focusing on the global path planning (Mac et al., 2016). The details of methods considered in this study are presented in following sections.

Potential Field (PF)

Potential Field (PF) considers the differences between two opposite forces, known as attraction and repulsion, to help a robot maneuvering within the environment (Khatib, 1985). In this approach, the next maneuvering location is determined based on how close the final destination is (attraction) and how strong the repulsive force of nearby obstacles are (Orozco-Rosas et al., 2019; Li et al., 2017; Mohammad et al., 2019). Since its first introduction by Khatib (1985), several modifications and variations of the PF algorithm been introduced to address it weaknesses such as getting trapped in local minima or being computationally intensive (MahmoudZadeh et al., 2018; Chen, 2017).

Song and Kumar introduced a decentralized PF-based control unit for directing and maneuvering multiple agents in scenarios where they are tasked to find goals collaboratively and trap and lead them towards a final destination (Peng Song and Kumar, 2003). One of the main concerns of PF is getting trapped in local minima. This issue has been addressed by Cheng and Zelinsky using temporary high magnitude attraction forces at a random position to take the robot out of local optima (Cheng and Zelinsky, 1995). Sfeir et al. (Sfeir et al., 2011) proposed using repelling force in order to reduce oscillations and reduce a chance of collision whenever the target is too close to obstacles. Sabatta and Siegwart proposed a hybrid PF that utilizes a vision-based component that computes PF cost function, assisting in determining the feasibility of various possible maneuvers (Sabatta and Siegwart, 2014).

Dijkstra's Algorithm (DA)

Edsger Dijkstra introduced Dijkstra's Algorithm (DA) in 1959, a classical algorithm that has proven its efficiency in finding the shortest possible path within a web of inter-connected nodes that represent spaces between obstacles (Dijkstra, 1959). DA method is known to be computationally intensive and being less effective if the distance between the starting location and the destination is far from each other (Noto and Sato, 2002). Noto and Soto (Noto and Sato, 2002) proposed an extended version of DA to solve this particular problem. DA main applications are in routing problems (Dijkstra, 1959; Risald et al., 2018; Broumi et al., 2017; Parungao et al., 2018).

Rapidly-Exploring Random Tree (RRT)

The Rapidly-Exploring Random Tree (RRT), introduced by LaValle and Kuffner Jr., is an algorithm based on stochastic search strategies (Kuffner and LaValle, 2002). The RRT is commonly applied in single query problems. Application of RRT in path planning is achieved by constructing a tree branching solution which revolves around random points selected within the searching space. In this approach, the starting point is considered as the root node for the RRT. Moving forward, a random point, selected based on the closest node construct will be identified. This step is repeated until the final destination is found and the outcome will form a feasible path which looks like a tree and covers almost all the empty space in the search space (Kuffner and LaValle, 2002; Connell and Manh La, 2018; Pimentel et al., 2018). This study utilised the variation of RRT, named RRT-based local path planning, as one of the base methods for its performance comparison. This variant uses the same concept and approach of global RRT with the main difference being lack of constructing a path first and execute it. In this variation, a random point or target point is chosen whilst the robot is moving depending on some probability calculated (Wong et al., 2018).

Probabilistic Road Map (PRM)

The main difference between Probabilistic Road Map (PRM) and RRT is in PRM's ability in multi-query planning. PRM was first introduced in 1996 by Kavraki et al. with their main goal being to provide a path for a robot in a static work-space (Kavraki et al., 1996). PRM generates a path between the initial and the destination positions by linking random nodes in the obstacle free area within the environment. The algorithm starts with selecting random points within the search area. The random points that fall on the

obstacle are neglected and only the random points in the obstacle free area are considered as viable nodes. All these nodes are connected from one to another to determine their feasibility path. Once all these nodes are connected, the path links that travel through an obstacle are categorized as infeasible and are abandoned. In the final stage, the shortest path between the initial location and the destination is identified from the remaining feasible paths (Kavraki et al., 1996; Sudhakara et al., 2018; Kumar et al., 2017).

Genetic Algorithm (GA)

Genetic Algorithm (GA), introduced by John Holland in 1975, is a search optimization algorithm based on the mechanics of the natural process (Holland et al., 1992). The fundamental principle of this approach is based on the survival of the strongest members of a population while the weaker members are abandoned. Each member of a population is define as a chromosome where each chromosome represents a possible solution for an optimization problem. The feasibility of the chromosomes are evaluated by measuring their fitness function. GA updates its population, strengthening its found solutions (chromosomes), by generating off-springs based on their best chromosomes through several processes or operators such as crossover and mutation. GA also has an elitism process that protects the best performing chromosome, fittest solution found (Holland et al., 1992; Corne and Lones, 2018; Kora and Yadlapalli, 2017).

Particle Swarm Optimization (PSO)

Kennedy and Eberhart introduced Particle Swarm Optimization (PSO) in 1995. PSO is a meta-heuristic algorithm inspired by swarm intelligence behaviors (e.g. bird flocking and fish schooling). This method defines members of a population as particles, possible solutions for a given problem, and it undergoes processes such as generation, evaluation, and update to help these particles to converge towards an optimal solution (Kennedy and Eberhart, 1995). In this algorithm, every particle's location in the search-space is represented by a position, X. PSO guides its solutions towards better regions in the search-spaces that have higher potential using a velocity equation, V (see equation 4). The best solution found by each particle is known as personal best ($PBest$) and the best solution found among the whole swarm is known as global best ($GBest$). PSO updates its particles' velocity through

following equations:

$$\begin{aligned} V_{i,j}(t) &= wV_{i,j}(t-1) + C_{i,j} + S_{i,j} \\ C_{i,j} &= c_1 r_{1,j} \times (PBest_{i,j}(t-1) - x_{i,j}(t-1)) \\ S_{i,j} &= c_2 r_{2,j} \times (GBest_{i,j}(t-1) - x_{i,j}(t-1)) \end{aligned} \tag{1}$$

In equation 1, $V_{i,j}(t)$ represents the velocity in iteration t. i and j represent the particle's index and the dimension in the search space respectively. c_1 and c_2 represent the acceleration coefficients of cognitive ($C_{i,j}$) and social ($S_{i,j}$) components respectively. $r_{1,j}$ and $r_{2,j}$ are random values in the range of [0,1] while w is the inertia weight that controls the influence of the last velocity in the updated version.

The efficiency of PSO's performance depends on how to adjust, control, and update its parameters. There are several parameter adjustment and controlling mechanisms suggested since PSO's introduction in 1995 aiming to enhance its overall performance. The conventional parameter adjustment and controlling mechanisms utilised to tune PSO optimization includes Fix Inertia Weight (FIW) (Aydilek et al., 2017; Raska and Ulrych, 2017), Linearly Decreasing Inertia Weight (LDIW) (Shi and Eberhart, 1998; Aydilek et al., 2017; Dai et al., 2018; Raska and Ulrych, 2017), Time Varying Acceleration Coefficient (TVAC) (Ratnaweera et al., 2004; Raska and Ulrych, 2017), Random Inertia Weight (RANDIW) (Ratnaweera et al., 2004; Aydilek et al., 2017; Dai et al., 2018; Raska and Ulrych, 2017), Random Acceleration Coefficients (RANDAC) (Suryanto et al., 2017), and Fix Acceleration Coefficients (FAC) (Ratnaweera et al., 2004; Aydilek et al., 2017; Dai et al., 2018; Raska and Ulrych, 2017). These parameter adjustment methods are all considered in this study. Equations 2 and 3 represent the LDIW and the TVAC formulations respectively.

$$w = (w_1 - w_2) \times \frac{(maxiter - t)}{maxiter} + w_2 \tag{2}$$

where, w_1 and w_2 are the initial and final inertia weights, t is the current iteration, and *maxiter* is the final iteration.

$$\begin{aligned} c_1 &= (c_{1f} - c_{1i}) \times \tfrac{t}{maxiter} + c_{1i} \\ c_2 &= (c_{2f} - c_{2i}) \times \tfrac{t}{maxiter} + c_{2i} \end{aligned} \tag{3}$$

where, c_{1f}, c_{1i}, c_{2f} and c_{2i} are the initial and final cognitive and social components, t is the current iteration, and *maxiter* is the termination iteration.

The equation for updating the particles is as follows:

$$x_{i,j}(t) = x_{i,j}(t-1) + V_{i,j}(t) \tag{4}$$

$PBest_{i,j}$ and $GBest_{i,j}$ represent the best solution found by the particle and the best overall solution found by the swarm and can be updated using equations 5 and 6. In these equations, f represents the fitness function utilised to assess the feasibility of the particle (x_i).

$$PBest_i(t) = \begin{cases} PBest_i(t-1), & \text{if } f(x_i(t)) \leq f(PBest_i(t-1)) \\ x_i(t), & \text{otherwise} \end{cases} \tag{5}$$

$$GBest(t) = argmin\left\{f(PBest_1(t)), ..., f(PBest_n(t))\right\} \tag{6}$$

Differential Evolution (DE)

Differential Evolution (DE) algorithm, introduced by Storn and Price in 1997, is another population-based approach with considerable similarities to GA due to its use of crossover, mutation, and selection operators (Storn and Price, 1997). The main difference between these two algorithms is that GA relies more on crossover operation while DE relies more on mutation operation. The mutation operator is the main search mechanism for DE as it takes advantage of the selection operation to direct the search towards high potential areas in the search space. In DE, target vectors, known as solution vectors, are generated and initialized in the first iteration and are improved by applying operators such as mutation, crossover and selection. The algorithm starts with random initialization of a population of solutions and ranks these solutions based on their fitness value acquired using an evaluation process. During the mutation process, the weighted difference of the two population vectors are added to the third vector resulting in generation of new parameter vectors (Piotrowski, 2017; Jain et al., 2018b). Mutation operation expands the search space and a mutant vector using following equation:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \tag{7}$$

where F is the scaling factor with value in the range of [0, 1] with solution vectors x_{r1}, x_{r2}, and x_{r3} being chosen randomly satisfying following conditions:

$$x_{r1}, x_{r2}, x_{r3} | r_1 \neq r_2 \neq r_3 \neq i, \tag{8}$$

where i is the index of current solution.

Crossover operation is used to increase the diversity of the disconcerted parameter vectors. The target vectors are mixed with mutated vectors and trial vectors are produced by:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{if } R_j \leq CR \\ x_{ji,G}, & \text{if } R_j > CR \end{cases} \tag{9}$$

where CR is a crossover constant and R_j is a random real number between [0, 1] with j denoting the j^{th} components of the resultant array. Subsequently, the resulting new vectors (trail vectors) compete with target vectors to determine the winner, which is brought forward to the next generation (Piotrowski, 2017; Jain et al., 2018b). The exploitation behaviour and the exploration behaviour depends on the scaling factor in equation 7. Exploration behavior occurs when the scaling factor is close to 1 and exploitation behaviour occurs with scaling factors close to 0.

Cuckoo Search Algorithm (CSA)

The Cuckoo Search Algorithm (CSA) is another meta-heuristic approach, introduced by Yang and Deb in 2009 (Yang and Deb, 2009). CSA is inspired from swarm intelligence and the behavior of the cuckoo species. CSA has three fundamental steps: i) each cuckoo is only allowed to lay one egg in each iteration and this egg is to be laid in a random nest based on Levy flights; ii) high quality eggs and nests are to be preserved for the next generation (elitism); and iii) the number of available host nests are fixed. In this method, the probability of egg discovery by the host bird of each nest that cuckoo laid egg in is $p_a \in [0, 1]$. This probability will determine the circumstances of either the host choosing to throw the egg away or abandon the nest and build a new one(Yang and Deb, 2009, 2014; Joshi et al., 2017). An extended version of the CSA algorithm introduced the case of each nest containing more than one egg (Yang and Deb, 2013). Equation 10 represents the levy flight necessary to generate a new solution $x(t + 1)$ for cuckoo indexed m.

$$x_m(t + 1) = x_m(t) + \partial \oplus Levy \tag{10}$$

where ∂ is the step size. It is common to use ∂=1 (Yang and Deb, 2009).

Equation 11 represents Levy Distribution formula used by equation 10 for large steps.

$$Levy \sim u = t^{-1-\beta}(0 < \beta < 2) \tag{11}$$

Initializing the step size (∂) with a large value and iteratively decreasing it results in convergence of the population towards a solution in the final generation. Yang (Yang and Deb, 2013) introduced an additional component to the equation 10 resulting in the following equation:

$$x_m(t+1) = x_m(t) + \partial \oplus Levy \sim 0.01 \frac{u}{\mid v \mid^{\frac{1}{\beta}}}(x_n(t) - x_m(t)) \tag{12}$$

where u and v are drawn from normal distribution which is

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_u^2) \tag{13}$$

where

$$\sigma_u = \{\frac{(\gamma(1+\beta)\sin(\frac{\pi\beta}{2}))}{(\gamma[(1+\beta)/2]\beta 2^{\frac{\beta-1}{2}})}\}^{\frac{1}{\beta}}, \sigma_v = 1 \tag{14}$$

γ is the standard gamma function (Joshi et al., 2017).

In equation 12 of CSA method, exploration occurs if a large difference between x_n and x_m is observed whilst a small difference between these two parameters promote exploitation. The advantages of CSA includes having fewer number of parameters to fine-tune and its feasibility to deal with multi-modal objective functions. CSA is reported to have insensitive convergence rate to p_a and there are cases where fine tuning of the parameters is not necessary at all (Yang and Deb, 2009, 2013, 2014). CSA is applied in various fields ranging from vehicle routing problem(Xiao et al., 2017), neural networks(Chatterjee et al., 2017), medical(Shehab et al., 2017), scheduling(Bibiks et al., 2018).

3. Experiments Setup

As stated earlier, this study aims to investigate feasibility of meta-heuristic based methods in robot motion planning problems. To do so, a collection of classical motion planning and navigation algorithms are selected to provide a performance base-line. These methods include Dijkstra's Algorithm (DA), Potential Field (PF), Rapid-exploring Random Tree (RRT) and Probabilistic Roadmap (PRM). In meta-heuristic category, a collection of well-known methods such as Genetic Algorithm (GA), Differential Evolution (DE), Cuckoo Search Algorithm (CSA) and five variations of Particle Swarm Optimization (PSO) algorithm are considered. The abbreviation and details of these PSO-based methods are as follows:

- Fix-PSO: Fix Inertia Weight (FIW) and Fix Acceleration Coefficients (FAC).
- Rand-PSO: Random Inertia Weight (RANDIW) and Random Acceleration Coefficients (RANDAC).
- TVAC-PSO: Linearly Decreasing Inertia Weight (LDIW) and Time Varying Acceleration Coefficient (TVAC).
- LDIWPSO: Linearly Decreasing Inertia Weight (LDIW) and Fix Acceleration Coefficients (FAC).
- CPSO: Constricted Particle Swarm Optimization.

It is noteworthy that the choice of meta-heuristic methods in this study, beside their successful ongoing implementations in robotics problems, is also based on the findings in (Ab Wahab et al., 2015) where these methods showed potential in benchmark problems. Two sets of experiments are considered in this study. Sub-Experiments that impose additional complexities and possible trajectories are introduced for further assessment of feasibility of motion planning methods utilised in this study. The scenarios are set to navigate a robot from a starting position towards a final destination within an obstacle-dense and hard to maneuver maze-like environment. Two sets of environment layouts are considered in these experiments mimicking maze and indoor navigation scenarios. Except for the DA method, neither classical nor meta-heuristic-based approaches are provided with any prior information about the location of obstacles or the environmental layouts. The pseudo code for Classical Path Planning Algorithms; Potential Field (Algorithm 1), Dijkstra's Algorithm (Algorithm 2), Rapidly exploring Random Tree (Algorithm 3), Probabilistic Road Map (Algorithm 4)) and Meta-heuristic Algorithms; Genetic Algorithm (Algorithm 5), Differential Evolution Algorithm (Algorithm 6), Particle Swarm Optimization Algorithm (Algorithm 7), and Cuckoo Search Algorithm (Algorithm 8) for local path planning are available in the Supplement Section.

Experiment I: **Maze Layout**

The Maze layout is illustrated in Fig.1. This layout is organized in a maze manner where the obstacles are scattered in such order that the path towards the destination is constantly encircled with obstacles for the robot. In this

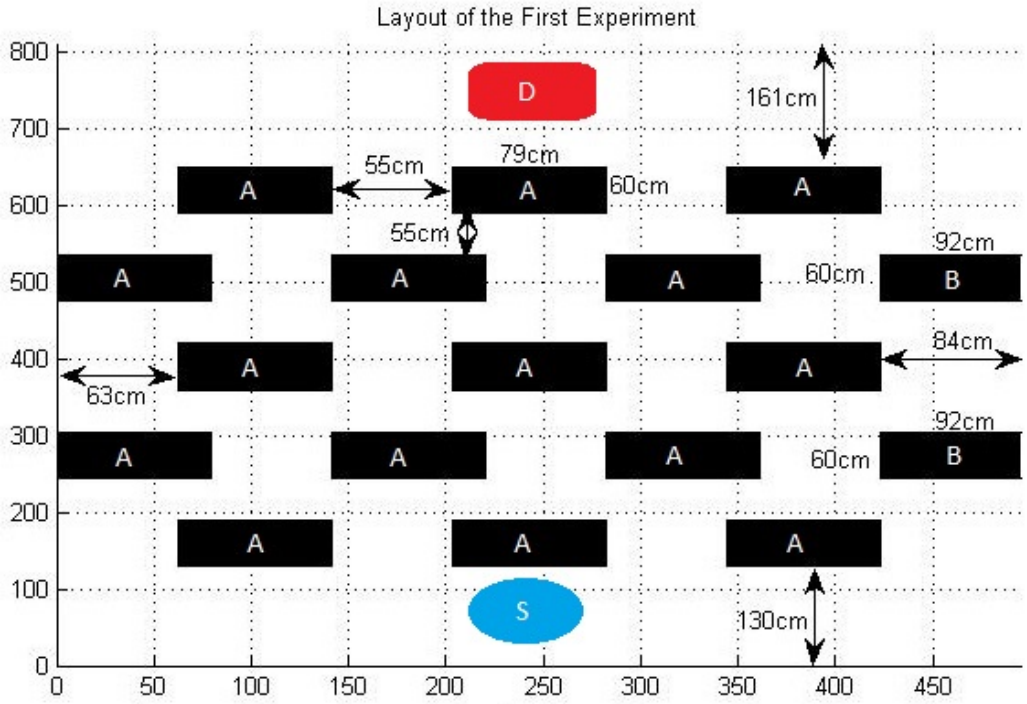


Figure 1: **Maze Environment Layout. Rectangles in black colour represent obstacles. The distance between two obstacles is set to less than twice the diameter of the robot navigating in the environment. Start and destination locations are marked as S and D and painted in blue and red colours respectively.**

layout, the starting location is marked with S (represented in Blue Oval Shape), the destination area is marked with D (represented in Red Hexagon Shape), and the obstacles are marked with A and B (represented in Black Rectangle Shapes). All obstacles have the same measurements (except for two obstacles that are marked as B). The dimensions of the maze layout is set to $8m \times 5m$. The dimensions of the obstacles marked as A are 79cm $\times$ 60cm and the other two obstacles, marked as B, are 92cm $\times$ 60cm. The gaps between obstacles are also consistent in most of the layout and are set in a manner to be less than twice of the size of the robot used (55cm). The gaps between the obstacles and the walls are set to 84cm on the right and 63cm on left. Fig.2 shows four overview of the maze layout from different angles from which the density of the obstacles can be observed.

Figure 2: **The view of Experiment I (Maze Layout) from four difference angles**

Experiment II: **Symmetric Layout**

Fig.3 illustrates the symmetric layout. Unlike the previous experiment (maze layout), this layout is designed to be symmetrical for both sides. It is also consists of irregular obstacles which have been placed in the middle of the environment setup. The environment is 6 meters long and 5 meters wide. This layout has a combination of one irregular-shaped obstacle with four sets of obstacles (mixed dimensions). The details of obstacles' measurements (length (l) $\times$ width (w)) is provided in Table 1. This layout utilizes two starting locations (represented as $S1$ and $S2$) and two destination locations (represented as $D1$ and $D2$). As a result, three sub-experiments are executed where the routes are i) $S1$ to $D1$, ii) $S2$ to $D1$ and iii) $S2$ to $D2$. The route between $S2$ to $D1$ is neglected due to its redundancy caused by the existing symmetry in this layout. Fig. 4 depicts four shots of the symmetric layout from different angles.

Platform

The Turtlebot with a dimension of 30 cm width is used in combination with Robotic Operating System (ROS) to carry out the experiments. Turtlebot is equipped with six infra-red sensors (acting as the main sensors) and

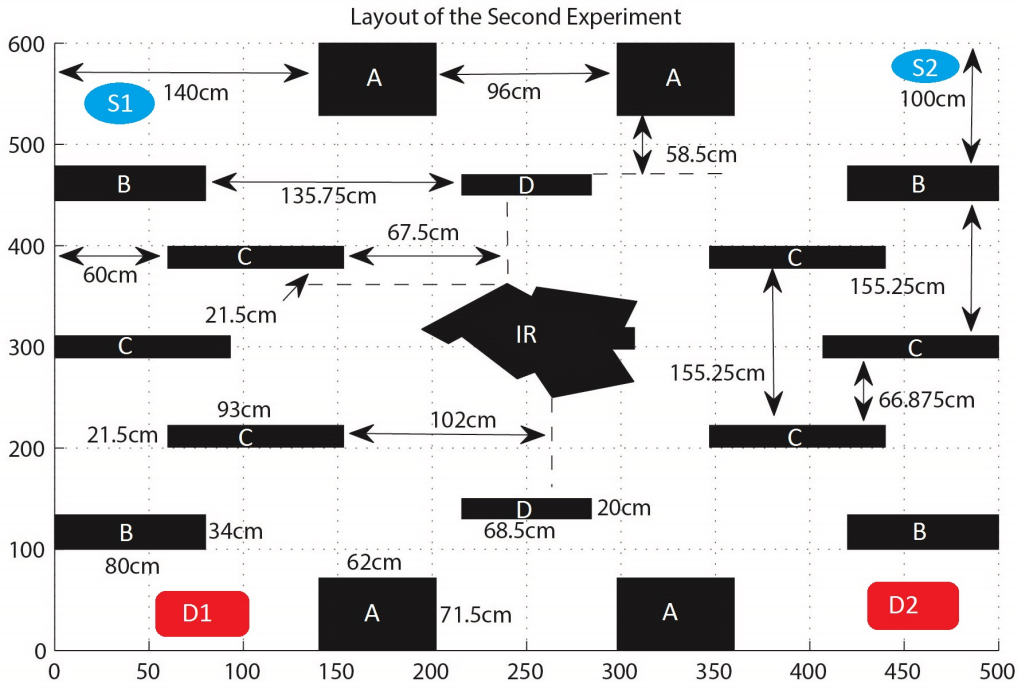


Figure 3: **Symmetric Environment Layout. Black rectangles represent obstacles with dimension of A is 62cm×71.5cm, B is 80cm×34cm, C is 93cm×21.5cm and D is 68.5cm×20cm while IR indicates Irregular shape of the obstacle. S1 and S2 represent the starting areas whilst D1 and D2 represent the destination areas.**

two bumpers. The robot is protected from having a high speed collision that may cause a large amount of error in the odometry sensor data by instructing the robot to reduce its speed immediately after infra-reds detects any object presents within its sensing range. Any collision that happens is considered as a controlled collision since the robot is programmed to stop immediately and perform a reverse mode to prevent the distortion of the odometry sensory data and accumulate unnecessary error in its odometry data. The recovery steps implemented in this robot once it encounters possible obstacle collision are i) select a temporary go-to location away from the obstacle location, ii) rotate towards the identified temporary go-to location, iii) move towards the direction of the temporary go-to location for 2 seconds, iv) return to motion planner algorithm to construct a new path towards destination location. Given that with exception of Dijkstra's algorithm, no information regard-

Figure 4: **The view of Experiment II (Symmetric Layout) from four difference angles**

ing the environmental layout and obstacle locations is provided to motion planning algorithms, the implemented obstacle collision mitigation method is necessary to protect the robot and its safe navigation.

Parameter Settings

Table 2 provides information regarding the parameter settings used in the study. Experiments I and II (consists of 3 Sub-Experiments) are repeated with 10 runs and the results acquired are averaged. A mobile robot (a Turtlebot) is assigned to maneuver through the obstacles scattered within

Table 1: **The measurements of the obstacles considered in the symmetric environment layout of experiment II**

Obstacle	Dimensions
A	$71.5cm \times 62.0cm$
B	$34.0cm \times 80.0cm$
C	$21.5cm \times 93.0cm$
D	$68.5cm \times 20.0cm$
IR	$552.5cm$ (parameter)

Table 2: **The settings for all algorithms involved in all Experiment**

Algorithm	**Settings**
PF	Laser Scan Minimum range set to 0.8 meter.
DA	Several unoccupied points between starting point and end point are defined before running the algorithm and full layout of the environment is presented to the algorithm.
RRT	Randomness point is set to 0.5.
PRM	100 random points are created for next moving point.
GA	Population size is set to 100 with 10% of population considered as best chromosomes in selection stage. Mutation rate is set 0.05% where mutation operation is to be selected if fitness is not improved in 5 consecutive iterations.
DE	Population size is set to 100 with crossover constant being set to 0.5
CSA	p_a is set to 0.25 and maximum number of nests are set to 100.
Fix-PSO	Inertia weight is set to 0.7 (FIW). Cognitive and social coefficients are set to 0.5 and 2.5 respectively (FIC).
Rand-PSO	Random values between 1.0 and 0.5 is used for inertia weight (RANDIW). Cognitive and social coefficients are set to random values number between 2.5 and 0.5 (RANDAC).
TVAC-PSO	Linear decreasing value is used for all inertia weight (LDIW) and Time Varying for Acceleration Coefficients (TVAC). Inertia weight is set to 0.9 as starting and 0.4 as end value. For social component, the acceleration coefficient is set to 2.5 as starting value and decreased to 0.5 with the respect of iteration while for cognitive component, the acceleration coefficient value is set to 0.5 initially and increased to 2.5 towards the end of iterations.
Linear-PSO	Linear decreasing approach varying from 0.9 to 0.4 is utilised for linear decreasing inertia weight (LDIW). Cognitive and social coefficient are set to fix values of 0.5 and 2.5 respectively (FAC).
CPSO	Cognitive and social coefficient are set to fix values of 0.5 and 2.5 respectively (FAC) with constricted value, K of 0.7299.

the environments from a fixed starting point towards a fixed destination location. From the classical motion planning methods discussed, Potential Field (PF), Dijkstra's algorithm (DA), Rapidly-exploring Random Tree (RRT) and Probabilistic Road Map (PRM) are considered, with the detailed information (full map) of the environment only being provided to DA while other methods are solely depending on their real-time sensory perception. Each node in the DA represents a centre point between two nearby obstacles and the distance between these nodes are considered as weights. As discussed earlier, from existing meta-heuristic methods, GA, DE, CSA and variations of PSO are considered. The parameter settings applied to the meta-heuristic methods are based on findings in (Ab Wahab et al., 2015).

4. Results and Discussion

To evaluate the performance of the selected algorithms in experiments I and II, seven performance factors are considered with Dijkstra's Algorithm (DA) is set as the benchmark of the best possible results that other algorithms

can achieved because it has full knowledge about the environment. These factors are:

1. **Ability to Arrive at Destination**: The ability of the algorithm to successfully direct the robot to the assigned destination based on odometry and stop the robot within 0.5m from the destination point. 0.5m is used due to the consideration of the size of the mobile robot and the location of the odometry which is in the middle of the robot.
2. **Number of Collisions**: The number of collisions happened during the executions.
3. **Displacement Problem**: If the robot's final position is out of the tolerance of the destination location, then it is considered as a *displacement problem*. The tolerance range is define as the radius around the goal location which is set to be one and a half size of the robot's radius. If the robot's final position is within the defined range, the trial is considered acceptable and the goal is considered reached.
4. **Execution Time**: The overall time taken by the robot to travel from the starting location to the destination position.
5. **Battery Consumption (%)**: The overall percentage of battery consumed by the robot to fully execute the task.
6. **Travelled Distance**: The overall distance travelled by the robot from the starting location to the destination position.
7. **Convergence Iteration**: This factor is considered to measure the numbers of iterations needed by meta-heuristic methods to find the destination location irrespective to the robot's ability to reach to that destination (in case of failed runs).
8. **Robot's Trajectory**: The trajectory is constructed based on the data produce by odometry sensor on the turtlebot. This factor shows the algorithm's consistency across multiple runs in terms of finding the path towards the destination defined.

Experiment I (Maze Layout)

The average results acquired from ten executions of motion planning algorithms in Experiment I are reported in Table 3. The results are discussed based on the factors introduced in previous section.

Considering the discussed performance factors, following observations are made:

1. **Ability to Arrive at Destination**, **Number of Collisions**, & **Displacement Problem**: All algorithms managed to drive the robot to its destination safely without any obstacle collision or displacement problems and within 0.5 meter radius of destination point.
2. **Execution Time (s)**: CPSO managed to outperform other algorithms including DA with 252.66 seconds on average across 10 executions. DA average execution time was 255.65 seconds, making it the second best performing algorithm. DE was the third performing algorithm with 257.44 seconds average execution time.
3. **Battery Consumption (%)**: CPSO outperformed other algorithms once again while only using 1.53% battery on average to complete the navigation task. DA, as the gold standard, had 1.55% battery consumption on average. The third best performing algorithm in this category is DE, utilizing 1.59% average battery consumption to travel the path it generated from start location to destination.
4. **Travelled Distance (m)**: DA achieved the best performance, an average travelled distance of 11.8966 meters. This performance is closely followed by CPSO with an average travelled distance of 12.0188 meters and Linear-PSO with average travelled distance of 12.1073 meters.
5. **Convergence Iteration**: This performance factor is only considered with meta-heuristic methods. CPSO out-performed other meta-heuristic methods with an average 484.5 iterations over 10 runs. DE and Linear-PSO were the 2^{nd} and 3^{rd} best performing methods with 502.0 and 531.90 average iterations respectively.

Fig. 5 illustrates the average performance differences from DA (e.g., the optimal global path planner method considered in this study) achieved by algorithms considered in this study. Given that DA had a prior knowledge of the Maze layout, its performance is considered as the benchmark performance that can be achieved by any method.

The results indicate that PF, DE, Fix-PSO, Rand-PSO and CPSO are the only methods with less than 5% difference compared to DA in execution time factor (Blue line in Fig. 5). Other methods shown more than 5% performance differences from DA with RRT and PRM showing more than 25% differences. This is due to RRT and PRM requiring initialization in their process once they are diverted from their initial path.

In energy consumption performance factor (Orange line in Fig. 5), PF, RRT, PRM and Fix-PSO achieved more than 10% performance differences

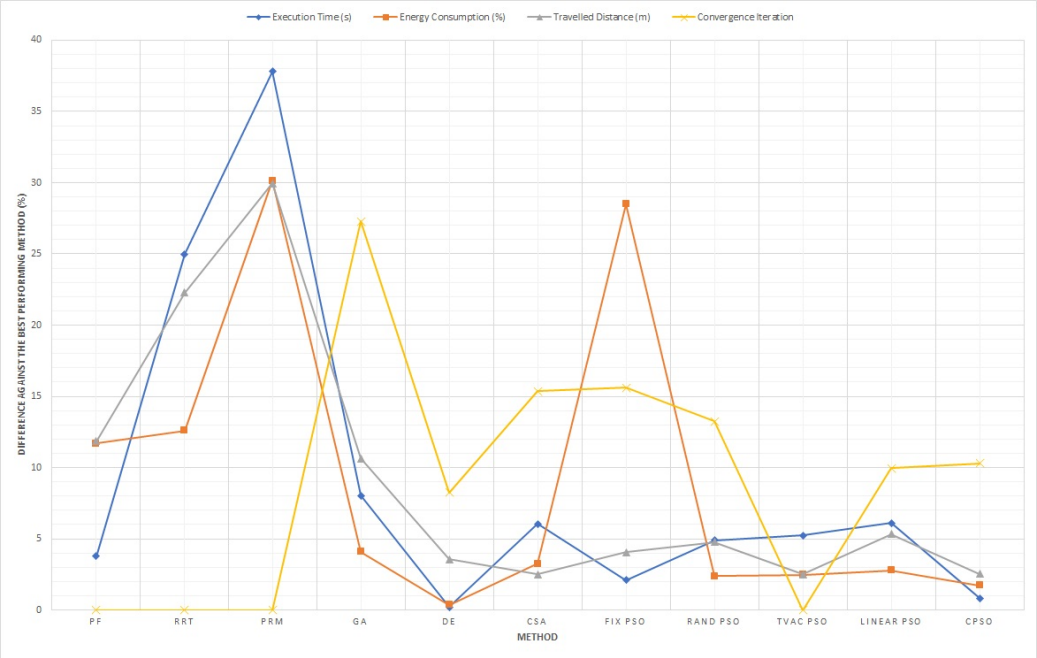


Figure 5: **Average performance difference from DA (optimal global path planner) achieved by various methods in Maze Layout.**

against DA (the best performing method). Other methods such as GA, DE, CSA, Rand-PSO, TVAC-PSO, Linear-PSO, and CPSO shown less than 5% differences compared to DA in this category.

Clear differences between classical and meta-heuristic methods are observed in travelled distance performance factor (Gray line in Fig. 5). PF, RRT and PRM shown major differences with DA, having more than 10% performance differences, while all meta-heuristic algorithms achieved better performances, less than 5% performance differences compared to the benchmark algorithm, DA. GA is an exception in meta-heuristic methods by achieving over 10% performance difference in travelled distance performance factor. The results in Fig. 5 indicate that TVAC-PSO is the best performing method between meta-heuristic approaches with GA performing worst than others (e.g., around 28% performance difference).

8. **Trajectory Traces**: Fig. 6 illustrated the trajectory traces for four selected methods (Benchmark - DA, Best - CPSO, Average - Rand-PSO, and Worst - PRM) in experiment I. Each sub-figure in Fig. 6

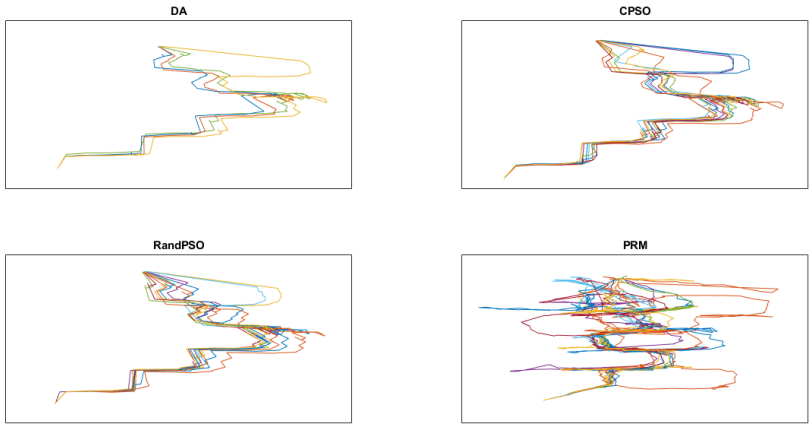


Figure 6: **Trajectory traces of four selected approaches (Benchmark - Top Left, Best - Top Right, Average - Bottom Left, and Worst - Bottom Right) in Experiment I. Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials. See complete results in Fig S1 in Supplement Material section.**

represents 10 paths travelled by the turtlebot robot (i.e. 10 executions of the algorithm) using an specific motion planning and navigation technique. These 10 routes in each sub-figure are illustrated using different colours. From the figure, DA is set as a benchmark and CPSO is considered as the best performing algorithm where the trajectory is more or less similar to DA with high consistency and precision. RandPSO represents the trajectory for an average performing algorithm and PRM represents the algorithm with the least favourable trajectory due to lack of consistency across executions/paths and low precision since almost none of the travelled paths are similar.

To better understand the causes of performance variations observed across approaches utilised, a unique representation of the robot's motion is used. In this representation, the Maze environment layout is divided into nine equal-sized rectangles (e.g., see Fig. 7), each called a region, and the average number of times (across 10 executions) that the robot is located in each region is counted. The results are presented using pie chart illustrations in Fig. 8. In this experiment, regions 2 and 9 contains starting and end points

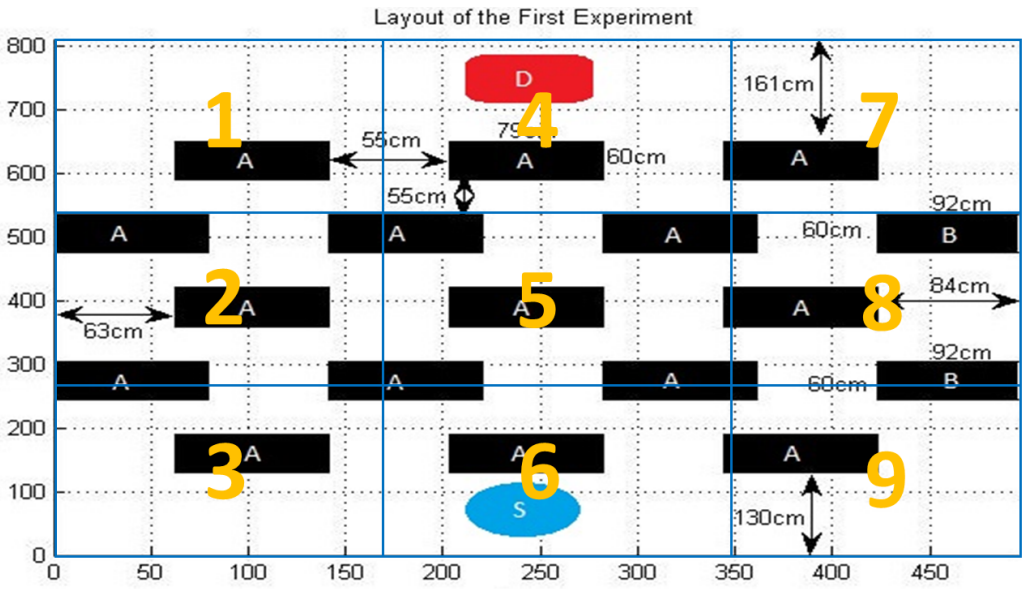


Figure 7: **Region classification on where the robot has been during the Experiment I**

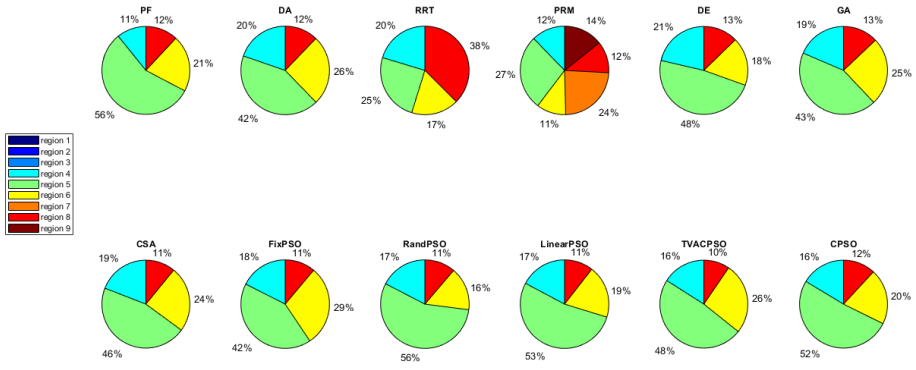


Figure 8: **Pie Chart of which region the robot has been during the Experiment I**

respectively.

It is noteworthy that DA, as the benchmark performing approach, only considered regions 4, 5, 6 and 8 in its trajectories with regions 5 and 8 having the highest and the least contributions in robot's chosen trajectory respectively. Given that regions 2, 5, and 8 are the regions containing three parallel rows of obstacles, these regions are likely to be the most difficult to manoeuvre in. This issue is reflected in the pie charts illustrated in Fig. 8 where the highest percentage is recorded in region 5. This is with exception of RRT algorithm in which region 8 received the highest percentage of occupation. It is also noticeable that in case of PRM algorithm the occupation percentage observed in region 6 of other methods is shared between regions 5 and 6. Unlike other approaches, PRM also included region 7 and 9 in its trajectories. The inclusion of these extra regions (7 and 9) are likely the reason behind the poor performances achieved by PRM algorithm in this experiment. RRT, unlike PRM, did not include any extra regions in its trajectory towards the destination. However, from the results depicted in the pie chart, it is noticeable that this approach spent an unusual amount of time in region 8. This is evident from allocating 38% of travelled path to region 8 by RRT in comparison with DE and GA in which the robot occupied this region in only 13% of its trajectory toward the destination. The best performing algorithm, CPSO, reported a smaller percentage (20%) for occupying region 6 compared with DA (26%). Furthermore, in comparison to DA, CPSO also reported a smaller percentage of travelled path in region 4 (destination region) while it performed poorly in region 5. Similar performance is observed by PF, recording 56% in average travelled path in region 5 while out-manoeuvring DA and CPSO in region 4 with 11% average occupation. The differences observed between CPSO and DA in their manoeuvrability in region 5 can be due to DA's advantage in terms of having full knowledge of the environment layout.

Experiment II (Symmetric Layout)

The results achieved from Experiment II are reported in Table 4. In order to have better understanding of how various approaches performed in this symmetric layout, the results are reported within four folds of Sub-Experiments and overall achievements. The results are discussed based on performance factors such as 1) *Ability to Arrive at Destination:*, 2) *Number of Collisions:*, 3) *Displacement Problem:*, 4) *Execution Time (s)*, 5) *Battery Consumption (%)*, 6) *Travelled Distance*, and 7) *Convergence Iteration (s).*

Sub-Experiment 1: Path from S1 to D1 in Fig. 3. The average results acquired from 10 executions of motion planning algorithms in Experiment II are reported in Table 4. The results are discussed based on the performance factors discussed earlier.

1. **Ability to Arrive at Destination**, **Number of Collisions** & **Displacement Problem**: Similar to previous experiments, Experiment I, neither collision nor displacement problems are observed in any of the executions with neither of the approaches utilised where all of them stop within 0.5 m radius from the destination point. A route between the starting and the destination locations is found by all algorithms.
2. **Execution Time (s)**: DA set the benchmark with an average execution time where it clocked only 87.55 seconds to complete the navigation task. CPSO and PF followed DA's performance in this category with 116.67 seconds and 126.76 seconds respectively.
3. **Battery Consumption (%)**: DA set the bar for other methods in battery consumption performance factor by only consuming 0.35% energy on average. CPSO's performance was the closest to the benchmark with 0.46% and followed by PF with 0.50% average energy consumption.
4. **Travelled Distance (m)**: CPSO and DE outperform DA with an average travelled distance of 7.1822 meters and 7.4456 meters respectively. DA only come as third best performing approach with 7.4706 meters. This is likely due to the fact that unlike DA that places its nodes in the middle point of two nearby obstacles (safe distance from either obstacle), CPSO and PF are allowed to risk colliding with obstacle by coming very close to the edges of the obstacles. Such advantage allows these methods to shorten their travelled distance while risking collision with obstacles.
5. **Convergence Iteration**: CPSO also outperformed others in the average convergence iteration category with 432.8 iterations. This performance is closely followed by DE with 449.1 iterations.

8. **Trajectory Traces**: Fig. 9 illustrates the trajectory traces for employed approaches in Sub-Experiment 1 in experiment II. Four subfigures in Fig. 9 represent 10 paths travelled by the turtlebot robot using an specific motion planning and navigation technique. These 10

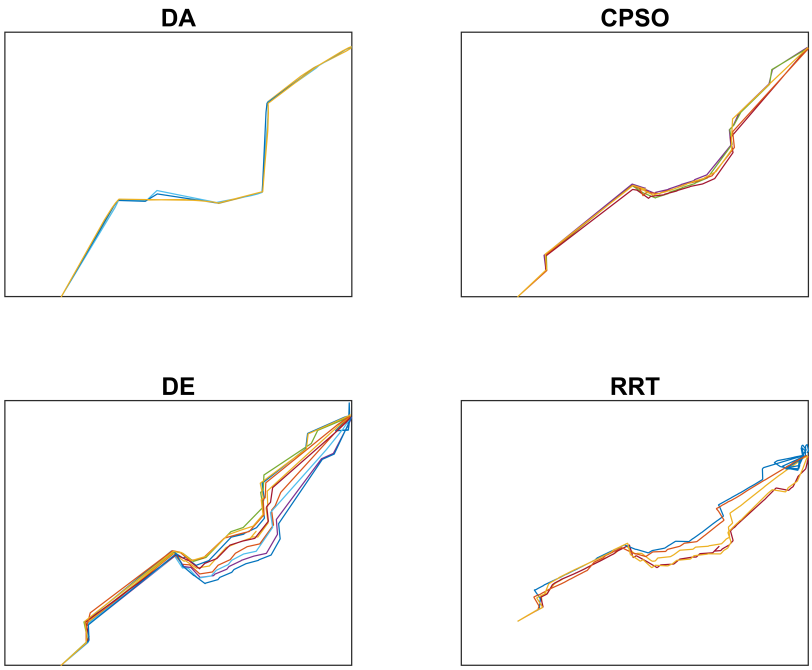


Figure 9: **Trajectory traces of four selected approaches (Benchmark - Top Left, Best - Top Right, Average - Bottom Left, and Worst - Bottom Right) in Experiment II (Sub-Experiment 1). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials. See complete results in Fig S2 in Supplement Material section.**

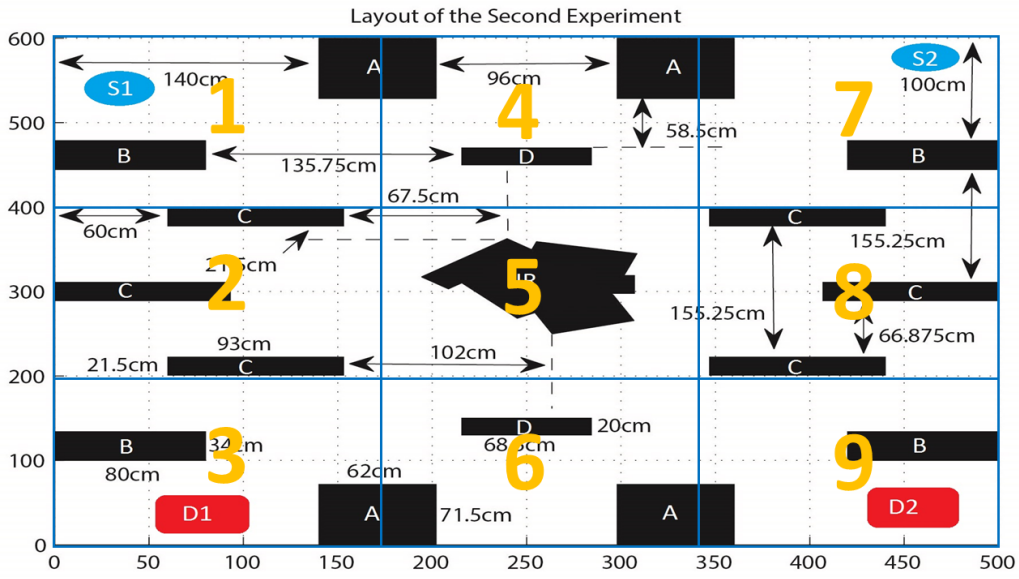


Figure 10: **Region classification on where the robot has been during the Experiment II**

routes in each sub-figure are illustrated using different colours. From the figure, DA is considered as the benchmark performance. CPSO, DE, and RRT have recorded same or similar travelled paths. CPSO is the best performing algorithm compared to DE and RRT. RRT is emerged as the least performing algorithm due to non-smooth trajectories travelled despite travelling similar paths to CPSO and DE.

Similar to experiment I, the layout of experiment II is divided into nine equal-sized rectangles named as regions (See Fig. 10). In Sub-Experiment 1, the starting and end points are located in regions 1 and 9 respectively with the irregular shape obstacle being placed in region 5 with partial coverage of region 8.

The average number of times (across ten executions) that the robot is located in each region is reported in the format of pie chart in Fig. 11. DA, as the benchmark performing method, spent (on average) 36% of its time within region 1, 26% within region 2 and 23% within region 5. DA also spent 11% of its time in destination region (region 9) and 4% in region 6. Unlike PRM and RRT who included region 4 in their trajectories, CPSO recorded similar regions as DA with major difference being in the sense of

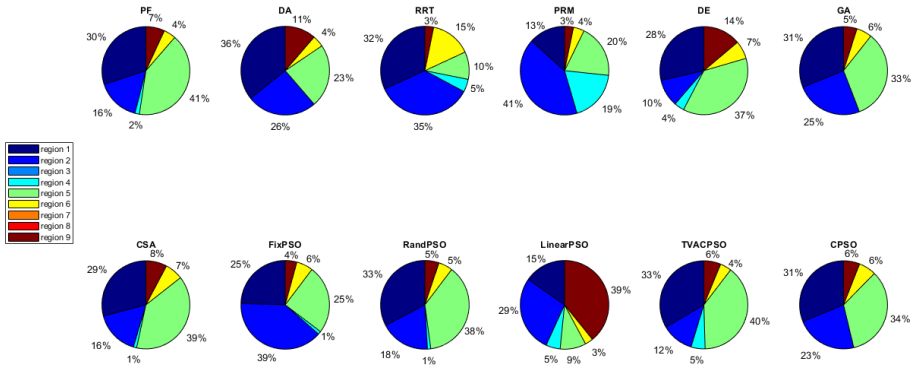


Figure 11: **Pie Chart of which region the robot has been during the Sub-Experiment 1 (Experiment II)**

the portion of the pie chart being allocated to region 5. It is noteworthy that CPSO manoeuvred slightly more efficiently compared to DA in regions 1, 2, 6 and 9. The marginal differences between DA and CPSO as observed in the pie charts is also supported by our findings reported in Table 4. The results indicate that the average travelled distance differences of these two techniques (in Sub-Experiment 1) are 0.29 meter. It is also noticeable that between all approaches, Linear-PSO reported the lest favourable performance in region 9 in this Sub-Experiment. This might be due to the robot's inability to find the exact location of the final destination resulting in several passage of the same route between already visited points in this region. This is reflected in the pie chart results with Linear-PSO spending maximum percentage of its travelled time in region 9 where the destination was located. Linear-PSO's 36% of travelled time spent in region 9 is considerably higher than DE's 14%. DE is ranked second in spending highest percentage of time in region 9.

Sub-Experiment 2: Path from S2 to D1 in Fig. 3 . The average results acquired from ten executions of motion planning algorithms in Experiment II are reported in Table 4. In this Sub-Experiment, DA outperformed all other motion planning methods in all performance factors. The results are discussed based on the factors introduced earlier.

1. **Ability to Arrive at Destination**, **Number of Collisions** & **Displacement Problem**: Similar to previous experiments, neither collision nor displacement problems are observed in any of the executions with neither of the approaches utilised where all of them stop within 0.5 m radius from the destination point.
2. **Execution Time (s)**: DA set the benchmark by clocking an average execution time of 77.03 seconds. This performance is followed by PF and CPSO with 106.66 and 115.98 seconds respectively.
3. **Battery Consumption (%)**: DA set the benchmark for other methods in average battery consumption performance factor with 0.31%. This performance is matched by CPSO who also achieve 0.31% average battery consumption. PF is the second best performing method in this performance factor by achieving 0.42% average battery consumption.
4. **Travelled Distance (m)**: CPSO and PF followed DA's 5.9697 meters average travelled distance with 6.1953 and 6.2051 meters respectively.
5. **Convergence Iteration**: CPSO outperformed other meta-heuristic methods in average convergence iterations performance factor with 370.0 iterations. This performance is closely followed by DE with 388.8 and Linear-PSO with 396.8 iterations.

6. **Trajectory Traces**: Fig. 12 depicts the trajectory traces of approaches employed in Sub-Experiment 2 of experiment II. In here, each sub-figure depicts ten paths, each presented using a unique colour, travelled by the turtlebot robot from the start to destination location. From the observations, DA is the benchmark for other algorithms by repeating its routes through 10 runs. Other algorithms such as PF - Best performer, TVAC-PSO - Average performer, and RRT - least favourable performer constructed almost similar routes. In this performance factor, after DA that had prior knowledge of environment layout, PF and RRT recorded the most and the least consistent paths across 10 executions respectively.

Fig. 13 provides pie chart representation of robot's manoeuvres in different regions in the experiment layout. In this Sub-Experiment, the initial and final positions of the robot are placed in regions 1 and 7 respectively. Irregular shape obstacle is located in region 5 with partial coverage of region 4.

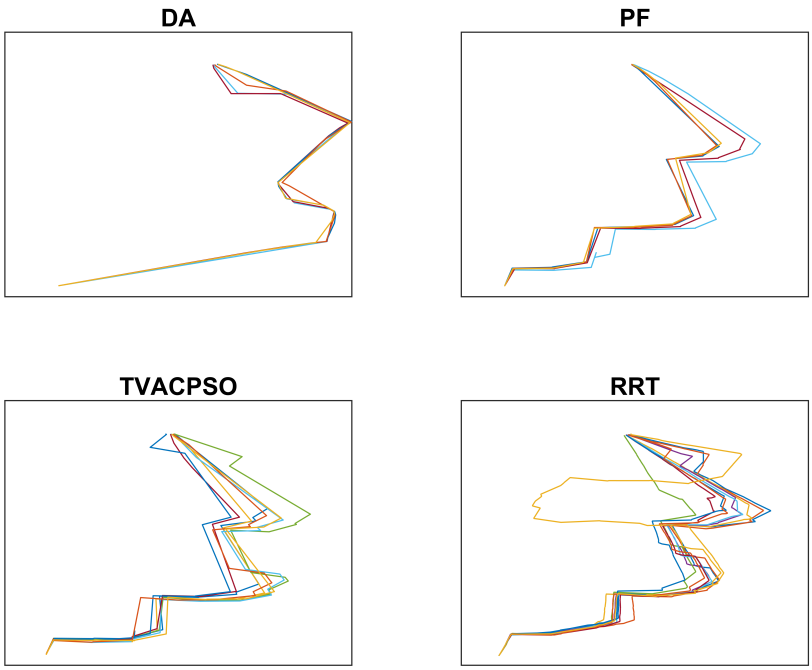


Figure 12: **Trajectory traces of four selected approaches (Benchmark - Top Left, Best - Top Right, Average - Bottom Left, and Worst - Bottom Right) in Experiment II (Sub-Experiment 2). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials. See complete results in Fig S3 in Supplement Material section.**

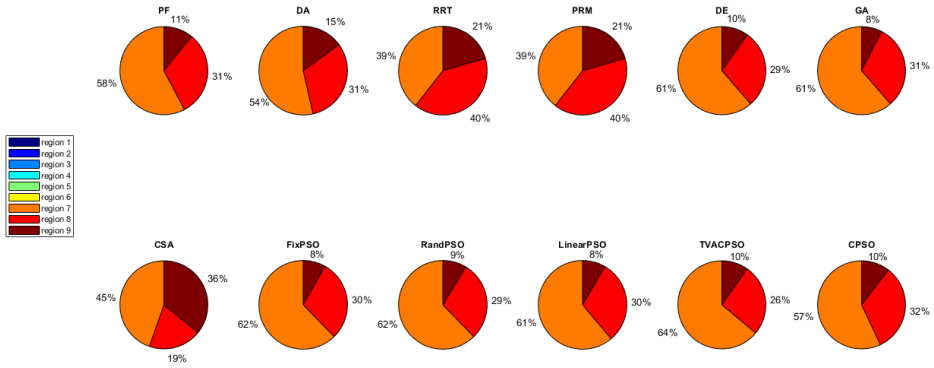


Figure 13: **Pie Chart of which region the robot has been during the Sub-Experiment 2 (Experiment II)**

Results presented in Fig. 13 indicate that all methods are consistent in their chosen trajectories by only visiting regions 7, 8, and 9 (see Fig. 10). This indicates that all methods are more or less using the same route. The main difference in the results is emerged as how various methods overcome the obstacle in region 8. From the results, CPSO and PF reported almost identical distributions across these three regions compared to the benchmark algorithm DA. The noticeable difference between the performances of these three algorithms is that DA has 4% higher average occupation of destination region while having around 4% less average occupation of region 1 compared to CPSO and PF. RRT and PRM demonstrated identical pie chart performances with 40%, 39% and 21% average time spent in regions 8, 7 and 9 respectively. This similarity is also reflected in their trajectory traces presented in Fig. 12 and similarities observed on various factors and categories reported in Table 4. GA and DE also reported similar average percentage of coverage distribution across the three regions. The most unique results are reported by CSA pie chart in which the highest average region occupation percentage is reported for the destination region (region 7). This issue better explains the poor performance of CSA reported in Table 4 in which in almost all categories it is identified as the worst performing approach.

Sub-Experiment 3: Path from S2 to D2 in Fig. 3. The travelled path in this sub-experiment is similar to of Sub-Experiment 1 (from S1 to D1 in Fig. 3). This sub-experiment is designed to investigate a different approaching angle of the irregular shape obstacle that is positioned in the center of the Experiment II environment layout. In here, the close-by sharp edges of the IR obstacle results in generating pseudo Cul-De-Sac type obstacles that in some degree increases the difficulty of path generation specially if the robot get trapped in such tight and hard to manoeuvre corners. The results are assessed based on the performance factors introduced earlier.

1. **Ability to Arrive at Destination**, **Number of Collisions** & **Displacement Problem**: Similar to previous findings in this environment, in this sub experiment, all algorithms managed to drive the robot to its destination safely without any obstacle collision or displacement problems and within 0.5 meter radius of destination point.
2. **Execution Time (s)**: DA benchmark this factor by clocking an average of 90.64 seconds execution time. This is considerably lower that CPSO and PF performances as second and third best performing methods in this performance factor. CPSO clocked an average execution time of 169.81 seconds and PF achieved an average execution time of 184.33s.
3. **Battery Consumption (%)**: DA benchmark this category with only 0.37% average battery consumption, followed by the CPSO and PF performances with 0.7031% and 0.7270% respectively.
4. **Travelled Distance (m)**: DA travelled the shortest path on average with 7.1247 meters followed by CPSO and PRM with 7.6538 and 7.6842 meters respectively.
5. **Convergence Iteration**: Rand-PSO emerged as the best performing algorithm with an average of 444.2 iterations. CPSO closely followed with an average performance of 453.9 iterations and GA as the third best performing algorithm for this category with an average of 462.0 iterations.

6. **Trajectory Traces**: The trajectory traces of the methods utilised in this experiment are illustrated in Fig. 14. Similar to the previous experiments, DA is consistent within its chosen path and considered as the benchmark results. CPSO also demonstrated consistency in its chosen path although it is not identical to DA. Rand-PSO is chosen in this

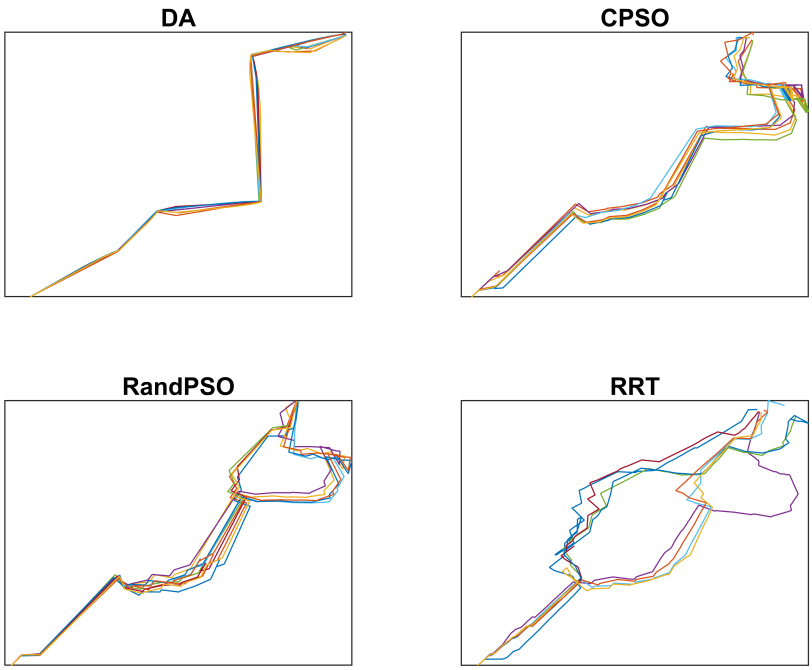


Figure 14: **Trajectory traces of four selected approaches (Benchmark - Top Left, Best - Top Right, Average - Bottom Left, and Worst - Bottom Right) in Experiment II (Sub-Experiment 3). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials. See complete results in Fig S4 in Supplement Material section.**

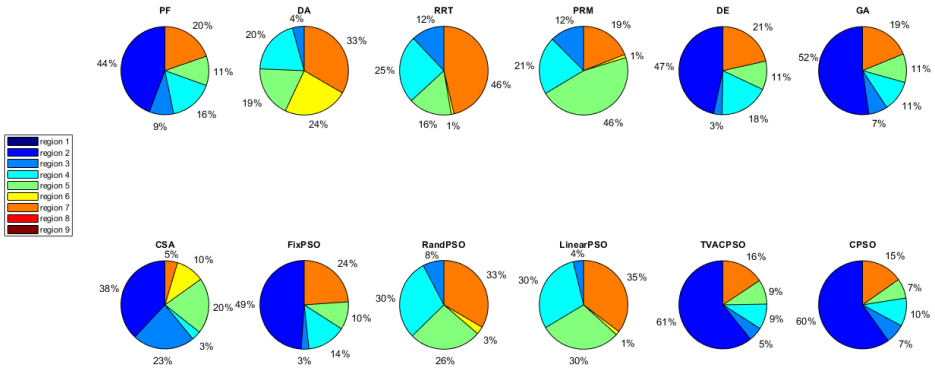


Figure 15: **Pie Chart of which region the robot has been during the Sub-Experiment 3 (Experiment II)**

illustration since it represents the average trajectory traces produced by other algorithms. RRT demonstrated a wandering behaviour near the destination point indicating their inability to identify the optimal path towards the destination. RRT trajectory traces indicated inconsistent behaviours with respect to the chosen manoeuvring strategies when they faced the irregular shape obstacle.

Fig. 15 depicts pie chart representation of robot's manoeuvres in different regions in this sub-experiment layout. In this sub-experiment, regions 7 and 3 contain the initial (S2) and the destination (D1) positions respectively with irregular shape obstacle being located in region 5. The results indicate that majority of the methods inhabited region 2 as part of their paths towards the destination. Considering DA as the benchmark performing approach, it is noteworthy that region 2 is ignored while the highest average occupation percentage for region 7 is observed. DA occupied region 7 by 33% on average while only RRT and Linear-PSO have reported higher average occupation percentage for this region. PRM and RRT demonstrated different region coverage, where RRT occupied region 7 longer compared to PRM and PRM occupied region 5 longer than all other methods. Unlike DA, RRT and PRM who excluded region 2 in their trajectories, CPSO recorded the second largest average occupation percentage for this region. Robot controlled by CPSO

performed exceptionally well in all regions apart from region 2. This is likely due to the robot experiencing Cul-De-Sac problem by being surrounded by several obstacles and not being able to find a clear way out resulting in achieving poor performances in this region. It is noticeable that the average travelled distance differences of CPSO is close to DA and superior to other conventional motion planing methods such as PRM. This is indicative that although CPSO performed poorly in region 2, but its exceptionally efficient performance in other regions (especially region 5 that contained the irregular shaped obstacle) resulted in the method becoming the second best performing approach in this sub-experiment.

Overall results across all sub-experiments in Experiment II: Considering the overall performances of all motion planning techniques across the three sub-experiments in this symmetric layout, CPSO is considered as the best performing approach. It should be noted that DA is considered the benchmark or the best result that other approaches can achieve since it has access to layout of the environment prior to conducting motion planning. PF and DE are the second and third best performing methods in this experiment due to their consistent high performances in most categories across all three sub-experiments in this layout.

Figs. 16, 17, 18 and 19 shows comparison of all methods in four categories of *Execution Time*, *Battery Consumption*, *Travelled Distance*, and *Convergence Iteration* respectively. These figures capture average performance information of all methods across all three Sub-Experiments. From the findings in Fig. 16, the Execution Time performance factor, DA is emerged as the best performing method and is considered the performance benchmark.

The results in Fig. 16 indicate that the shape of the graph for each sub-experiment is almost identical from one to another. Only PF and CPSO have approximately 5% performance different compared with DA in this performance measure factor with DE closely following DA's performance with approximately 5% difference in every sub-experiment. The worst performing method in this performance measure factor is RRT where all sub-experiments have at least 15% performance difference compared to DA.

In Energy Consumption category (see Fig. 17), PF and CPSO are performing similarly to the best performing method, e.g. DA, in all sub-experiments by consuming less than 5% energy compared to DA. The least favourable performing approach in this category is Fix-PSO where its overall performance is more than 20% different from DA.

Fig.18 demonstrate the differences between the performance of DA and

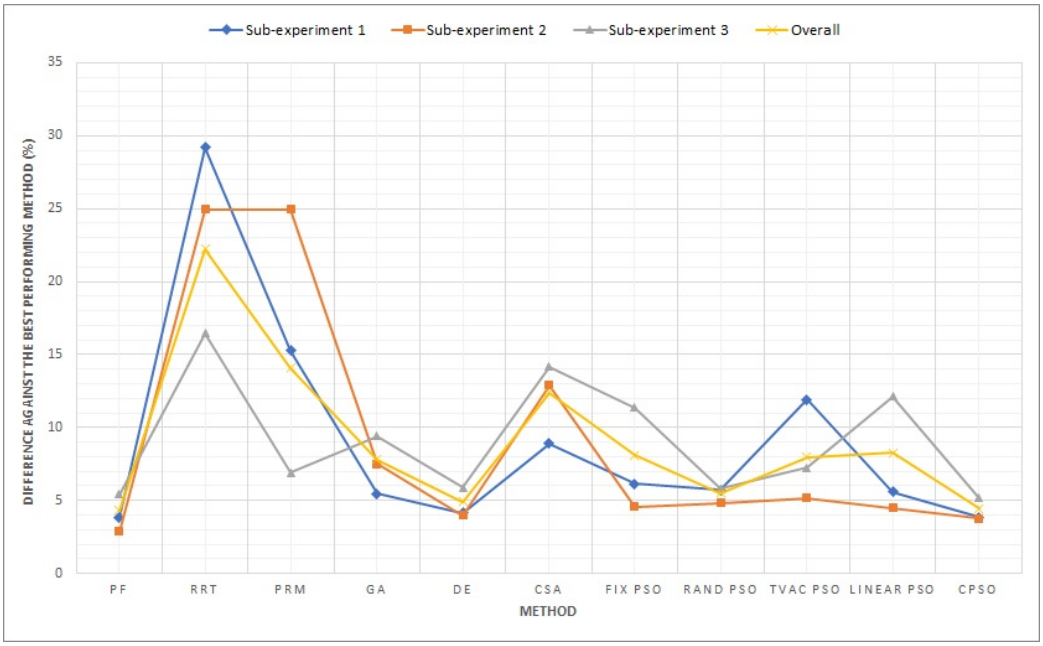


Figure 16: **Average performance difference from DA (optimal global path planner) in Sub-Experiments of Experiment II (Execution Time Factor)**

all other methods in Travelled Distance category. Since Travelled Distance influences the Execution Time and Battery Consumption performance factors, it is considered as the the main factor in this study. Given that DA's performance is considered the optimal performance (e.g. benchmark performance) in this study, deviation from it is considered as a measure of efficiency in motion planning methods utilised in the study. CPSO is considered as the best method given that the difference between CPSO and DA is less than 5% for each sub-experiment. RRT and PRM can be considered as the worst performing algorithms in this category as their differences are more than 5% across all Sub-Experiment. Within variants of PSO considered in this study, TVAC is found to be the worst performing variant since it has never achieved less than 5% performance difference from DA.

Since Convergence Iteration factor is considered for meta-heuristic algorithms only, the classical methods are marked with 0% in the Fig.19. The best performing approach in this performance factor is Rand PSO with other methods demonstrating approximate average convergence iteration differences of 10% compared to Rand PSO. CSA is the least successful approach in

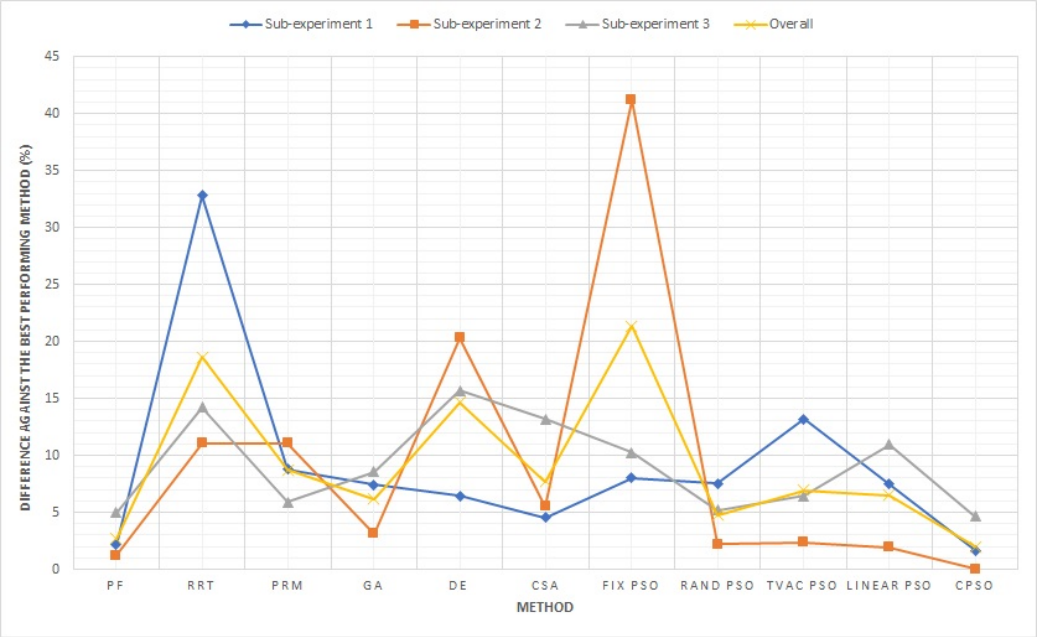


Figure 17: **Average performance difference from DA (optimal global path planner) in Sub-Experiment of Experiment II (Energy Consumption Factor)**

this category with over 30% average convergence iteration in sub-experiments 1 and 2. CSA recorded the highest convergence iteration in sub-experiment 3.

Comparing classical and heuristic-based methods, heuristic-based motion planning methods performed decently especially in experiment II where all heuristic-based approaches (with the exception of Fix-PSO) managed to outperform all classical algorithms (with the exception of DA) in all categories considered. However, in experiment II, PF managed to outperform all meta-heuristic algorithms in most of the categories considered. It is also worth mentioning that RRT and PRM performed poorly in terms of execution time and travelled distance. Although, these methods, e.g. RRT and PRM, are popular navigation and motion planning approaches among classical methods, they only perform well if they are implemented under global path planning condition (Blanco et al., 2015; Pan and Manocha, 2016; Wang et al., 2018; Contreras-Cruz et al., 2015; Masehian and Sedighizadeh, 2010; Mohanta and Keshari, 2019; Masehian and Sedighizadeh, 2013). In classical motion planning methods, PF outperformed RRT and PRM in all categories.

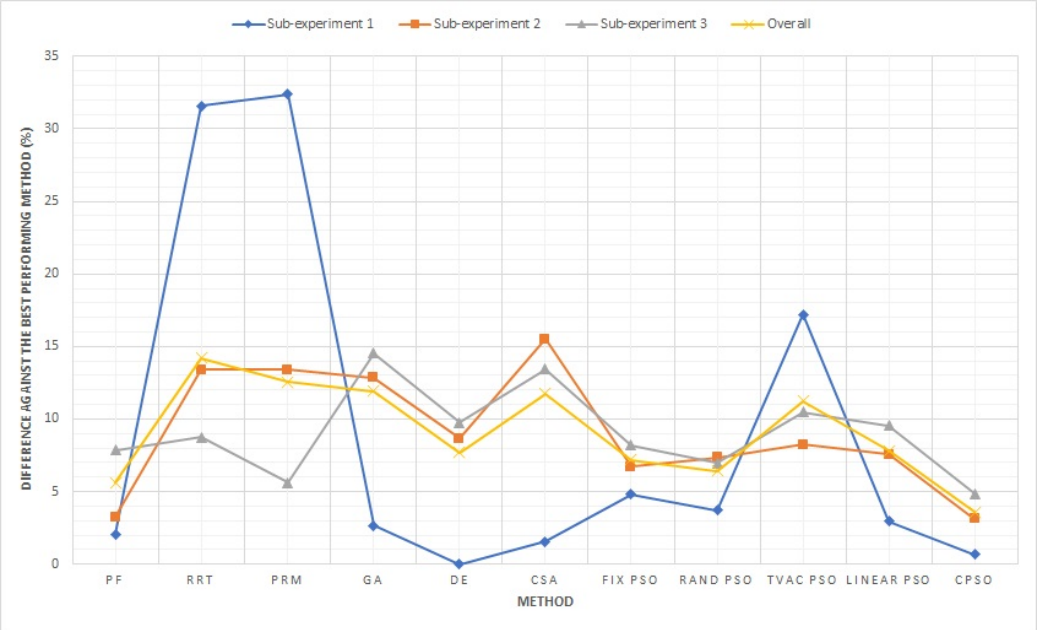


Figure 18: **Average performance difference from DA (optimal global path planner) in Sub-Experiments of Experiment II (Travelled Distance Factor)**

This is because PF is more direct and do not get influenced from random factors as in the case of RRT and PRM.

Although DA managed to outperform other methods in all relevant categories, it is noteworthy that DA has an advantage compared to others. Unlike other methods that have no knowledge about the environment layout which force them to perform local motion planning on the basis of their real-time sensory readings, DA had access to the complete environmental layout and been constantly performing global path planning. Therefore, the results achieved by DA is considered as the absolute optimal performance. From the overall results, it can be concluded that CPSO is the best performing approach given that it became the second best to DA in almost every category considered in the study.

Constriction factor, K, in CPSO has given an influence in reducing the velocity of the particles in search environment which as a result limited the range of fluctuations of particles in search space, resulting in faster convergence towards optimal solutions (Wang et al., 2018; Jain et al., 2018a; Clerc, 2011). This crucial step has helped CPSO to construct better local paths

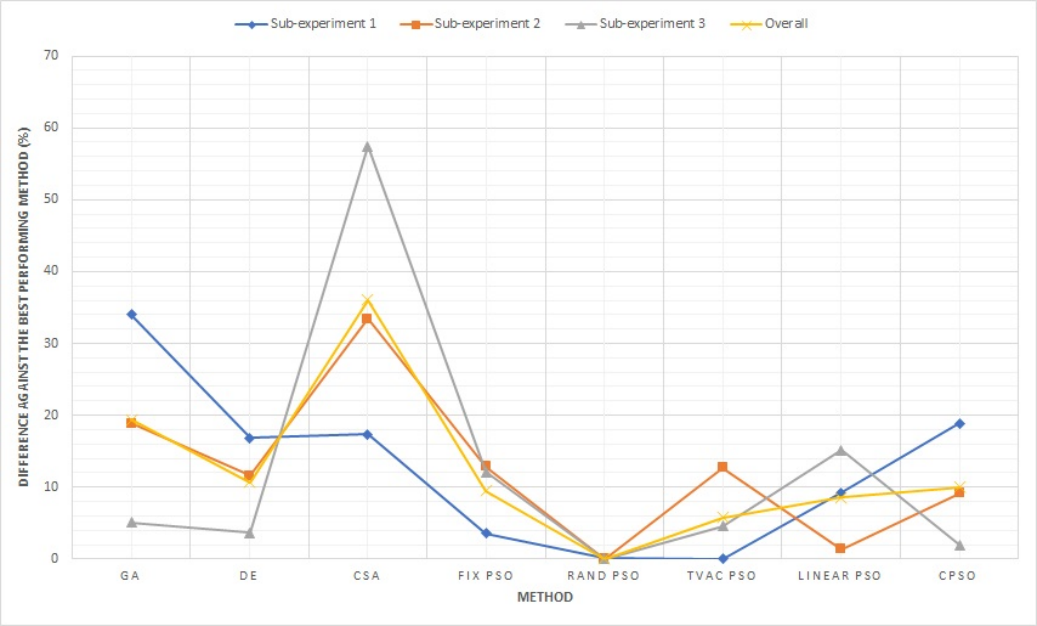


Figure 19: **Average performance difference from the best performing algorithm in Sub-Experiments of Experiment II (Convergence Iteration Factor)**

which are shorter and less congested with obstacles compared to the other meta-heuristic methods. Our study concludes that CPSO is the most sufficient motion planning approach for local path planning problems. This further supports Eberhart and Shi statement in (Eberhart and Y.shi, 2000) where they explained the important role of the constriction factor in the particle swarm performance by stating "*the best approach to use with particle swarm optimization as a "rule of thumb" is to utilize the constriction factor approach*".

5. Conclusion

This study is focused on providing a comparison between conventional motion planning approaches such as Potential Field (PF), Dijkstra's Algorithm (DA), Rapidly-explore Random Tree (RRT), Probabilistic Road Map (PRM) and a collection of meta-heuristic based methods such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Cuckoo Search (CSA). The performance of meta-heuristic based methods are first evaluated under a collection of benchmark functions. These

methods are chosen based on the findings and recommendations of (Ab Wahab et al., 2015) where using a collection of benchmark functions, variations of PSO (Linear-PSO, Rand-PSO and CPSO) and DE outperformed other meta-heuristic methods. Two sets of experiments featuring a maze and a symmetric layout are designed to further evaluate the feasibility of these meta-heuristic methods under motion planning problems. The layouts are designed to be hard-to-manoeuvre and obstacle-rich.

The results indicated DA as the benchmark performing method with understanding that this approach had access to prior knowledge of the environment and a pre-planned path whilst other techniques only had information about their starting location (S) and their target destination (D). Therefore, DA's performance is considered as the control or benchmark result and its performance is considered as the best possible outcome that can be achieved in these experimental layouts. The summary of the first, second, and third best performing algorithms for all experiments are presented in Table 6. The results points to CPSO as the best performing algorithm by outperforming all other meta-heuristic based and conventional motion planning approaches investigated in this study. CPSO is found to be among the top 3 performing algorithms across all experiments and performance measure factors considered in this study. CPSO outperformed all other methods in 2 experiments under execution time, Battery Consumption and consistency of trajectory trace performance factors. CPSO outperformed other methods in 3 experiments in Travelled Distance performance factor. The main drawback of this method is observed under Convergence Iteration performance factor where it only been in top 3 performing algorithms for one of the experiments and been outperformed by other meta-heuristic methods under this performance measure factor.

It is noteworthy that no statistical significance is found between CPSO and DA under travelled distance category in any of the motion planning experiments. Such lack of significant difference between CPSO and DA is also observed in the execution times and battery consumption categories. The consistent competitive performance of CPSO under benchmark functions problems are also reported in (Ab Wahab et al., 2015). CPSO's superior performance observed under the assessed experimental motion planning problems investigated in this study testifies on its efficiency and suitability for local path planning and navigation problems. This is an emerging field that its solutions can be embedded in any autonomous system and problem domain such as driver-less cars and autonomous robotic systems.

Acknowledgment

This project is a collaboration between Robot, Computer Vision, and Image Processing (RCVIP) research group from School of Computer Sciences, Universiti Sains Malaysia, Autonomous System and Advanced Robotics (ASAR) Research Lab, the University of Salford, and NeuroCognition Laboratory (NCL) at University of Colorado Colorado Spring. This project was supported by Universiti Sains Malaysia Short Term Grant (PKOMP/6315262).

Table 3: **Averaged results achieved by various approaches in Experiment I (Maze Layout). BOLD fonts represent the best performing method in each performance factor.**

Factor	**PF**	**DA**	**RRT**	**PRM**	**GA**	**DE**	**CSA**	**Fix PSO**	**Rand PSO**	**TVAC PSO**	**Linear PSO**	**CPSO**
Arrived at Destination(Times)	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Number of Collisions (Times)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Displacement Problem (Times)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Execution Time (seconds)	287.57	**255.65**	466.70	574.94	323.42	257.44	306.69	273.30	296.95	299.83	307.33	262.66
Battery Consumption (%)	2.77	**1.55**	2.86	4.68	1.98	1.59	1.89	4.51	1.80	1.80	1.84	1.73
Travelled Distance (m)	12.8936	**11.8966**	13.7719	14.4165	12.7922	12.1954	12.1075	12.2378	12.2994	12.1073	12.3441	12.0188
Convergence Iteration	—	—	—	—	565.3	542.0	550.7	551.0	548.1	**531.9**	544.1	544.5
Best Performing Algorithm	3	**6**	3	3	3	3	3	3	3	4	3	3

Table 4: **Averaged results achieved by various approaches utilised in Experiment II. The best result achieved in each category within each Sub-Experiment is reflected using BOLD font.**

Factor	**PF**	**DA**	**RRT**	**PRM**	**GA**	**DE**	**CSA**	**Fix PSO**	**Rand-PSO**	**TVAC-PSO**	**Linear-PSO**	**CPSO**
Experiment II Sub-Experiment 1: Path from S1 to D1 in Fig. 3												
Arrived at Destination	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Number of Collisions	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Displacement Problem	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Execution Time (s)	126.7588	**87.5493**	385.1796	243.1524	143.1296	129.8887	178.1093	150.0594	145.7491	208.9403	144.3843	126.6746
Battery Consumption (%)	0.4970	**0.3524**	2.5631	0.9421	0.8531	0.7864	0.6565	0.8939	0.8605	1.2389	0.8568	0.4637
Travelled Distance (m)	7.5604	7.4706	9.2181	9.2613	7.5950	**7.4456**	7.5327	7.7145	7.6536	8.4111	7.6117	7.4822
Convergence Iteration	—	—	—	—	470.6	519.6	472.0	432.8	422.9	**422.7**	449.1	476.5
Best Performing Algorithm	3	**5**	3	3	3	4	3	3	3	4	3	3
Experiment II Sub-Experiment 2: Path from S2 to D1 in Fig. 3												
Arrived at Destination	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Number of Collisions	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Displacement Problem	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Execution Time (s)	106.6591	**77.0312**	334.8004	334.5404	154.5190	118.0599	210.0471	124.4046	126.8047	130.4547	123.4652	115.9784
Battery Consumption (%)	0.4154	**0.3115**	1.2537	1.2537	0.5786	2.0363	0.7826	3.8131	0.4970	0.5119	0.4748	0.3131
Travelled Distance (m)	6.2051	**5.9697**	6.9449	6.9433	6.9026	6.5982	7.0973	6.4567	6.5045	6.5692	6.5190	6.1953
Convergence Iteration	—	—	—	—	543.2	488.8	652.2	497.8	**401.5**	496.8	411.5	470.0
Best Performing Algorithm	3	**6**	3	3	3	3	3	3	4	3	3	3

Table 5: **Averaged results achieved by various approaches utilised in Experiment II. The best result achieved in each category within each Sub-Experiment is reflected using BOLD font. Table 4 continued...**

Factor	**PF**	**DA**	**RRT**	**PRM**	**GA**	**DE**	**CSA**	**Fix-PSO**	**Rand-PSO**	**TVAC-PSO**	**Linear-PSO**	**CPSO**
Experiment II Sub-Experiment 3: Path from S2 to D2 in Fig. 3												
Arrived at Destination	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Number of Collisions	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Displacement Problem	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Execution Time (s)	184.3313	**90.6391**	374.3786	209.6193	252.7201	192.5292	335.1722	286.9791	190.9152	215.9695	299.9560	179.8085
Battery Consumption (%)	0.7270	**0.3709**	1.3910	0.7937	0.9830	1.4948	1.3168	1.1053	0.7419	0.8309	1.1573	0.7031
traveled Distance (m)	8.4680	**7.1247**	8.6183	8.0842	9.6089	8.7962	9.4214	8.5221	8.3127	8.9183	8.7546	7.6538
Convergence Iteration	—	—	—	—	462.0	468.9	723.5	502.9	**444.2**	466.3	517.9	453.9
Best Performing Algorithm	3	**6**	3	3	3	3	3	3	4	3	3	3
Experiment II Overall												
Arrived at Destination	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
Number of Collisions	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Displacement Problem	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Execution Time (s)	139.2497	**85.0732**	364.7862	262.4374	183.4562	146.8259	241.1095	187.1477	154.4897	185.1215	189.2685	140.8205
Battery Consumption(%)	0.5465	**0.3449**	1.7359	0.9965	1.4392	0.8049	0.9186	1.9374	0.6998	0.8605	0.8296	0.4933
Travelled Distance (m)	7.4111	**6.8550**	8.2604	8.0963	8.0355	7.6133	8.0171	7.5644	7.4903	7.9662	7.6284	7.2104
Convergence Iteration	—	—	—	—	473.8	510.6	615.9	477.8	**422.9**	461.9	459.5	466.8
Best Performing Algorithm	9	**17**	9	9	9	11	9	9	10	10	9	9

Table 6: Summary of the First, Second, and Third Best Performing Algorithm in Each Experiments

Factor	Exp. I	Exp. II(Sub-1)	Exp. II(Sub-2)	Exp. II (Sub-3)
Ability to Arrive at Destination	All algorithms perform adequately in these performance measures.			
Number of Collisions				
Displacement Problem				
Execution Time	1 - DE, 2 - CPSO, 3 - PF	1 - CPSO, 2 - PF, 3 - DE	1 - PF, 2 - CPSO, 3 - DE	1 - CPSO, 2 - PF, 3 - Rand-PSO
Battery Consumption	1 - DE, 2 - CPSO, 3 - Rand & TVAC-PSO	1 - CPSO, 2 - PF, 3 - CSA	1 - PF, 2 - CPSO, 3 - Rand-PSO	1 - CPSO, 2 - PF, 3 - Rand-PSO
Travelled Distance	1 - CPSO, 2 - TVAC-PSO, 3 - Fix-PSO	1 - DE, 2 - CPSO, 3 - CSA	1 - CPSO, 2 - PF, 3 - Linear-PSO	1 - CPSO, 2 - PF, 3 - PRM
Convergence Iteration	1 - TVAC-PSO, 2 - DE, 3 - Linear-PSO	1 - Rand-PSO 2 - Fix-PSO, 3 - Fix-PSO	1 - Rand-PSO 2 - Linear-PSO, 3 - DE	1 - Rand-PSO 2 - CPSO, 3 - GA
Trajectory Traces	1 - Linear-PSO 2 - CPSO, 3 - Fix-PSO	1 - CPSO 2 - PF, 3 - DE	1 - PF 2 - CPSO, 3 - Fix-PSO	1 - CPSO 2 - PF, 3 - DE

References

References

Ab Wahab, M.N., Nefti-Meziani, S., Atyabi, A., 2015. A comprehensive review of swarm optimization algorithms. PLoS ONE doi:10.1371/journal.pone.0122827.

Atyabi, A., Phon-Amnuaisuk, S., Ho, C.K., 2010. Navigating a robotic swarm in an uncharted 2D landscape. Applied Soft Computing Journal doi:10.1016/j.asoc.2009.06.017.

Atyabi, A., Powers, D.M.W., 2013. Review of classical and heuristic-based navigation and path planning approaches. International Journal of Advancements in Computing Technologies .

Aydilek, I.B., Nacar, M.A., GüMüŞÇü, A., Salur, M.U., 2017. Comparing inertia weights of particle swarm optimization in multimodal functions, in: IDAP 2017 - International Artificial Intelligence and Data Processing Symposium. doi:10.1109/IDAP.2017.8090225.

Bayat, F., Najafinia, S., Aliyari, M., 2018. Mobile robots path planning: Electrostatic potential field approach. Expert Systems with Applications doi:10.1016/j.eswa.2018.01.050.

Bibiks, K., Hu, Y.F., Li, J.P., Pillai, P., Smith, A., 2018. Improved discrete cuckoo search for the resource-constrained project scheduling problem. Applied Soft Computing Journal doi:10.1016/j.asoc.2018.04.047.

Blanco, J.L., Bellone, M., Gimenez-Fernandez, A., 2015. TP-space RRT - Kinematic path planning of non-holonomic any-shape vehicles. International Journal of Advanced Robotic Systems doi:10.5772/60463.

Broumi, S., Talea, M., Bakali, A., Smarandache, F., 2017. Application of Dijkstra algorithm for solving interval valued neutrosophic shortest path problem, in: 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016. doi:10.1109/SSCI.2016.7850151.

Chatterjee, S., Dey, N., Sen, S., Ashour, A.S., Fong, S.J., Shi, F., 2017. Modified cuckoo search ased neural networks for forest types classification, in: Frontiers in Artificial Intelligence and Applications. doi:10.3233/978-1-61499-785-6-490.

Chen, J., 2017. Optimal path planning of robot based on ant colony algorithm. Acta Technica CSAV (Ceskoslovensk Akademie Ved) .

Chen, L., 2014. UUV path planning algorithm based on virtual obstacle, in: 2014 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2014. doi:10.1109/ICMA.2014.6885960.

Cheng, G., Zelinsky, A., 1995. A Physically Grounded Search in a Behaviour Based Robot, in: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence.

Clerc, M., 2011. Beyond Standard Particle Swarm Optimisation. International Journal of Swarm Intelligence Research doi:10.4018/jsir.2010100103.

Connell, D., Manh La, H., 2018. Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots. International Journal of Advanced Robotic Systems doi:10.1177/1729881418773874.

Contreras-Cruz, M.A., Ayala-Ramirez, V., Hernandez-Belmonte, U.H., 2015. Mobile robot path planning using artificial bee colony and evolutionary programming. Applied Soft Computing Journal doi:10.1016/j.asoc.2015.01.067.

Corne, D., Lones, M.A., 2018. Evolutionary algorithms, in: Handbook of Heuristics. doi:10.1007/978-3-319-07124-4_27.

Dai, H.P., Chen, D.D., Zheng, Z.S., 2018. Effects of Random Values for Particle Swarm Optimization Algorithm. Algorithms doi:10.3390/a11020023.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik doi:10.1007/BF01386390.

Duguleana, M., Mogan, G., 2016. Neural networks based reinforcement learning for mobile robots obstacle avoidance. Expert Systems with Applications doi:10.1016/j.eswa.2016.06.021.

Eberhart, R.C., Y.shi, 2000. Comparing Inertia Weight and Constriction Factor in PSO. Pardue School of Engineering and Technology doi:10.3233/KES-2010-0211.

Elbanhawi, M., Simic, M., 2014. Sampling-based robot motion planning: A review. doi:10.1109/ACCESS.2014.2302442.

Freund, E., Kaever, P., 2017. Autonomous Mobile Robots. IFAC Proceedings Volumes doi:10.1016/s1474-6670(17)54619-4.

Holland, J.H., Chu, E.C., Beasley, J.E., 1992. Genetic Algorithms - Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. Scientific American .

Jain, N.K., Nangia, U., Jain, J., 2018a. A Review of Particle Swarm Optimization. doi:10.1007/s40031-018-0323-y.

Jain, S., Kumar, S., Sharma, V.K., Sharma, H., 2018b. Improved differential evolution algorithm, in: 2017 International Conference on Infocom Technologies and Unmanned Systems: Trends and Future Directions, ICTUS 2017. doi:10.1109/ICTUS.2017.8286085.

Joshi, A.S., Kulkarni, O., Kakandikar, G.M., Nandedkar, V.M., 2017. Cuckoo Search Optimization- A Review, in: Materials Today: Proceedings. doi:10.1016/j.matpr.2017.07.055.

Kamil, F., Hong, T.S., Khaksar, W., Moghrabiah, M.Y., Zulkifli, N., Ahmad, S.A., 2017. New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. Expert Systems with Applications doi:10.1016/j.eswa.2017.05.059.

Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation doi:10.1109/70.508439.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. Natural Computing Series doi:10.1007/978-3-319-93073-2_6, arXiv:9780201398298.

Khatib, O., 1985. Real-time obstacle avoidance for manipulators and mobile robots, in: Proceedings - IEEE International Conference on Robotics and Automation. doi:10.1109/ROBOT.1985.1087247.

Kora, P., Yadlapalli, P., 2017. Crossover Operators in Genetic Algorithms: A Review. International Journal of Computer Applications doi:10.5120/ijca2017913370.

Kuffner, J., LaValle, S., 2002. RRT-connect: An efficient approach to single-query path planning. doi:10.1109/robot.2000.844730.

Kumar, N., Vamossy, Z., Szabo-Resch, Z.M., 2017. Robot path pursuit using probabilistic roadmap, in: CINTI 2016 - 17th IEEE International Symposium on Computational Intelligence and Informatics: Proceedings. doi:10.1109/CINTI.2016.7846393.

Lamini, C., Benhlima, S., Elbekri, A., 2018. Genetic algorithm based approach for autonomous mobile robot path planning, in: Procedia Computer Science. doi:10.1016/j.procs.2018.01.113.

Li, W., Yang, C., Jiang, Y., Liu, X., Su, C.Y., 2017. Motion Planning for Omnidirectional Wheeled Mobile Robot by Potential Field Method. Journal of Advanced Transportation doi:10.1155/2017/4961383.

Mac, T.T., Copot, C., Tran, D.T., Keyser], R.D., 2016. Heuristic approaches in robot path planning: A survey. Robotics and Autonomous Systems 86, 13 – 28. URL: http://www.sciencedirect.com/science/article/pii/S0921889015300671, doi:https://doi.org/10.1016/j.robot.2016.08.001.

MahmoudZadeh, S., M.W Powers, D., Sammut, K., Atyabi, A., Yazdani, A., 2018. A hierarchal planning framework for AUV mission management in a spatiotemporal varying ocean. Computers and Electrical Engineering doi:10.1016/j.compeleceng.2017.12.035.

Masehian, E., Sedighizadeh, D., 2010. Multi-objective robot motion planning using a particle swarm optimization model. Journal of Zhejiang University SCIENCE C doi:10.1631/jzus.c0910525.

Masehian, E., Sedighizadeh, D., 2013. An Improved Particle Swarm Optimization Method for Motion Planning of Multiple Robots. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 175–188. URL: https://doi.org/10.1007/978-3-642-32723-0_13, doi:10.1007/978-3-642-32723-0_13.

Mohammad, S., Rostami, H., Sangaiah, A.K., Wang, J., 2019. Obstacle avoidance of mobile robots using modified potential field algorithm. Eurasip Journal on Wireless Communications and Networking .

Mohanan, M.G., Salgoankar, A., 2018. A survey of robotic motion planning in dynamic environments. doi:10.1016/j.robot.2017.10.011.

Mohanta, J.C., Keshari, A., 2019. A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. Applied Soft Computing Journal doi:10.1016/j.asoc.2019.03.055.

Nazarahari, M., Khanmirza, E., Doostie, S., 2019. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications doi:10.1016/j.eswa.2018.08.008.

Noto, M., Sato, H., 2002. A method for the shortest path search by extended Dijkstra algorithm. doi:10.1109/icsmc.2000.886462.

Orozco-Rosas, U., Montiel, O., Sepúlveda, R., 2019. Mobile robot path planning using membrane evolutionary artificial potential field. Applied Soft Computing Journal doi:10.1016/j.asoc.2019.01.036.

Pan, J., Manocha, D., 2016. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. International Journal of Robotics Research doi:10.1177/0278364916640908.

Parungao, L., Hein, F., Lim, W., 2018. Dijkstra algorithm based intelligent path planning with topological map and wireless communication. ARPN Journal of Engineering and Applied Sciences .

Patle, B.K., Babu L, G., Pandey, A., Parhi, D.R., Jagadeesh, A., 2019. A review: On path planning strategies for navigation of mobile robot. doi:10.1016/j.dt.2019.04.011.

Peng Song, Kumar, V., 2003. A potential field based approach to multi-robot manipulation. doi:10.1109/robot.2002.1014709.

Pennock, G.R., 2005. Robot motion: Planning and control. Mechanism and Machine Theory doi:10.1016/0094-114x(86)90035-2.

Pimentel, J.M., Alvim, M.S., Campos, M.F., Macharet, D.G., 2018. Information-Driven Rapidly-Exploring Random Tree for Efficient Environment Exploration. Journal of Intelligent and Robotic Systems: Theory and Applications doi:10.1007/s10846-017-0709-0.

Piotrowski, A.P., 2017. Review of Differential Evolution population size. Swarm and Evolutionary Computation doi:10.1016/j.swevo.2016.05.003.

Raska, P., Ulrych, Z., 2017. Testing different particle swarm optimization strategies, in: Proceedings of the 30th International Business Information Management Association Conference, IBIMA 2017 - Vision 2020: Sustainable Economic development, Innovation Management, and Global Growth.

Ratnaweera, A., Halgamuge, S.K., Watson, H.C., 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation doi:10.1109/TEVC.2004.826071.

Risald, Mirino, A.E., Suyoto, 2018. Best routes selection using Dijkstra and Floyd-Warshall algorithm, in: Proceedings of the 11th International Conference on Information and Communication Technology and System, ICTS 2017. doi:10.1109/ICTS.2017.8265662.

Sabatta, D., Siegwart, R., 2014. Bearings-only path following with a vision-based potential field, in: IEEE International Conference on Intelligent Robots and Systems. doi:10.1109/IROS.2014.6942939.

Sathiya, V., Chinnadurai, M., 2019. Evolutionary Algorithms-Based Multi-Objective Optimal Mobile Robot Trajectory Planning. Robotica doi:10.1017/S026357471800156X.

Sfeir, J., Saad, M., Saliah-Hassane, H., 2011. An improved Artificial Potential Field approach to real-time mobile robot path planning in an unknown environment, in: ROSE 2011 - IEEE International Symposium on Robotic and Sensors Environments, Proceedings. doi:10.1109/ROSE.2011.6058518.

Shehab, M., Khader, A.T., Al-Betar, M.A., 2017. A survey on applications and variants of the cuckoo search algorithm. doi:10.1016/j.asoc.2017.02.034.

Shi, Y., Eberhart, R.C., 1998. Parameter selection in particle swarm optimization. In Evolutionary Programming. doi:10.1007/978-3-319-46173-1.

Storn, R., Price, K., 1997. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization doi:10.1023/A:1008202821328.

Sudhakara, P., Ganapathy, V., Sundaran, K., 2018. Probabilistic roadmaps-spline based trajectory planning for wheeled mobile robot, in: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017. doi:10.1109/ICECDS.2017.8390129.

Suryanto, N., Ikuta, C., Pramadihanto, D., 2017. Multi-group particle swarm optimization with random redistribution, in: Proceedings - International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2017. doi:10.1109/KCIC.2017.8228445.

Wang, D., Tan, D., Liu, L., 2018. Particle swarm optimization algorithm: an overview. Soft Computing doi:10.1007/s00500-016-2474-6.

Wong, C., Yang, E., Yan, X.T., Gu, D., 2018. Optimal Path Planning Based on a Multi-Tree T-RRT Approach for Robotic Task Planning in Continuous Cost Spaces, in: Proceedings - 2018 12th France-Japan and 10th Europe-Asia Congress on Mechatronics, Mecatronics 2018. doi:10.1109/MECATRONICS.2018.8495886.

Xiao, L., Hajjam-El-Hassani, A., Dridi, M., 2017. An application of extended cuckoo search to vehicle routing problem, in: 2017 International Colloquium on Logistics and Supply Chain Management: Competitiveness and Innovation in Automobile and Aeronautics Industries, LOGISTIQUA 2017. doi:10.1109/LOGISTIQUA.2017.7962869.

Yang, X.S., Deb, S., 2009. Cuckoo search via Levy flights, in: 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings. doi:10.1109/NABIC.2009.5393690.

Yang, X.S., Deb, S., 2013. Multiobjective cuckoo search for design optimization. Computers and Operations Research doi:10.1016/j.cor.2011.09.026.

Yang, X.S., Deb, S., 2014. Cuckoo search: Recent advances and applications. doi:10.1007/s00521-013-1367-1.

Zafar, M.N., Mohanta, J.C., 2018. Methodology for Path Planning and Optimization of Mobile Robots: A Review, in: Procedia Computer Science. doi:10.1016/j.procs.2018.07.018.

Zammit, C., Van Kampen, E.J., 2018. Comparison between A* and RRT Algorithms for UAV Path Planning. doi:10.2514/6.2018-1846.

Supplemental Materials: A Comparative Review on Mobile Robot Path Planning: Classical or Meta-heuristic Methods?

Here are the Pseudo-code for all approaches involved in the experiments set.

Algorithm 1: Potential Field (PF)

Input : Laser scan range divided into 5 direction $(far\_left, left, middle, right, far\_right)$
Output: Fittest direction

```
for all possible direction, calculate fitness do
    if path is clear then
        move towards fittest direction
    else
        move to a random point
    end if
end for
return travel path
```

Algorithm 2: Dijkstra's Algorithm (DA)

Input : A list coordinate points x_i, y_i, $i = 1, 2, \cdots, n$, where each element is a point free from any obstacle.
Output: Shortest distance between connected points.

```
for all points selected do
    if path is clear then
        execute the movement
    else
        move to a random point → move towards original path
        created
    end if
end for
return travel_path
```

Algorithm 3: Rapidly-exploring Random Tree (RRT)

Input : A list of possible coordinate points x_i, y_i, $i = 1, 2, \cdots, n$.
Output: The closest point towards the destination.

while *Distance* $(q_{new}, q_{goal} > d_{threshold})$ **do**
    $q_{target} = Random\_Node()$
    $q_{nearest} = T.Nearest\_Neighbor(q_{target})$
    $q_{new} = Extend(q_{nearest}, q_{target}, expansion\_time)$
    if($q_{new}! = NULL$) $q_{new}.setParent(q_{nearest})$ $T.add(q_{new})$
     $ResultingPoint \leftarrow T.TraceBack(q_{new})$
end while
for *the ResultingPoint created* **do**
    selected the point as the destination
    if *path is clear* **then**
        execute the movement
    else
        move to a random point
    end if
end for
return *travel path*

Algorithm 4: Probabilistic Road Map (PRM)

Input : A list of possible coordinate points x_i, y_i, $i = 1, 2, \cdots, n$.
Output: The closest point towards the destination.

```
G(V,E) = NULL //Initialize a graph as empty
iteration_limit = n //number of nodes to make graph out of
Rad = r //radius of neighborhoods
for iteration < iteration_limit) do
    Xnew = RandomPosition()
    Xnearest = Near(G(V,E), Xnew, Rad) //find all nodes within a
     Rad
    Xnearest = sort(Xnearest) //sort by increasing distance
    for node in Xnearest do
        if not ConnectedComp(Xnew,node) & not
         Obstacle(Xnew,node) then
            G(V,E) += {Xnew,node} //add edge and node to graph
             Xnew.comp += node.comp//add Xnew to connected
             component
        end if
    end for
end for
Return G(V,E)
for the G(V,E) selected do
    set the point as the destination
    if path is clear then
        execute the movement
    else
        move to a random point
    end if
end for
return travel path
```

Algorithm 5: Genetic Algorithm (GA)

Input : The parameter setting for the Genetic Algorithm
Output: The fittest point towards the destination.

Use GA algorithm bellow to generate possible points
Initiate population with random points in solution space
Evaluate all chromosomes in the population
while *Maximum Iteration is not reached or fitness_solution* $\leq target\_fitness$ **do**
    Replace the fittest_solution with the fittest chromosome if it is fitter than fittest_solution
    Perform Selection Process eliminating 50% of chromosomes that have lower fitness
    Perform Crossover Process to repopulate the population
    Evaluate all chromosomes in the Population
    Increase the current iteration by 1
end while
for *all points created* **do**
    select the fittest point
    if *path is clear* **then**
        execute the movement
    else
        move to a random point
    end if
end for
return *travel path*

Algorithm 6: Differential Evolution Algorithm (DE)

Input: The parameter setting for the Differential Evolution Algorithm

Output: The fittest point towards the destination.

Use DE algorithm bellow to generate possible points

Initiate population with random points in solution space

Evaluate all agents in the population

while *Maximum Iteration is not reached or fitness_solution* $\leq target\_fitness$ **do**

  For each agent,j in the population do:

    Choose three agents a, b, and c that is, $1 \leq a, b, c, \leq N$ *with* $a \neq b \neq c \neq j$.

    Generate a random integer $i_{rand} \in (1, N)$.

    For each parameter i

    $y^{i,g} = x^{a,g} + F(x^{b,g} - x^{c,g})$

    $z_j^{i,g} = \left\{ \begin{matrix} y_j^{i,g} if rand() \leq CR or j = j_{rand}) \\ x_j^{i,g} otherwise \end{matrix} \right\}$

    End For

    Replace $x^{i,g}$ *with the child* $z^{i,g}$ *if* $z^{i,g}$ *is better*

    End For

    Increase the current iteration by 1

end while

for *all points created* **do**

  select the fittest point

  if *path is clear* **then**

    execute the movement

  else

    move to a random point

  end if

end for

return *travel path*

Algorithm 7: Particle Swarm Optimization Algorithm (PSO)

Input : The parameter setting for the Particle Swarm Optimization Algorithm

Output: The fittest point towards the destination.

Use PSO algorithm bellow to generate possible points

Initiate population with random points in solution space

Initiate personal best (pbest) and global best (gbest) solutions with random points in solution space

Evaluate all particles in the population, all their pbests and the swarm gbest

while *Maximum Iteration is not reached or gbest_fitness* $\leq target\_fitness$ **do**

- *Update the pbest for each particle in the swarm if the current particle is fitter*
- *Update the gbest with the fittest pbest if it is fitter than gbest*
- *Update the Velocity for each particle in the swarm using following equation* $V_{new} = W \times V_{old} + r_1 \times c_1(pbest - x_{old}) + r_2 \times c_2(gbest - x_{old})$
- *Update each particle in the population with its new velocity using following equation* $x_{new} = V_{new} + x_{old}$
- *Evaluate all particles in the Population, all their pbests and the swarm gbest*
- *Increase the current iteration by 1*

end while

for *all points created* **do**

- *select the fittest point (gbest)*
- **if** *path is clear* **then**
  - *execute the movement*
- **else**
  - *move to a random point*
- **end if**

end for

return *travel path*

Algorithm 8: Cuckoo Search Algorithm (CSA)

Input : The parameter setting for the Cuckoo Search Algorithm
Output: The fittest point towards the destination.

Use CSA algorithm bellow to generate possible points
Creates objective function $f(x)$ to evaluate fitness of possible points
Generate initial population of n host nest
Evaluate fitness and rank eggs
while *Maximum Iteration is not reached or Stop criterion* **do**
    $t = t + 1$
    Get a cuckoo randomly/generate new solution by Levy flights
    Evaluate quality/fitness, F_i
    Choose a random nest j
    if $F_i > F_j$ **then**
        Replace j by the new solution
    end if
    Worst nest is abandoned with probability P_a and new nest is built
end while
for *all points created* **do**
    select the fittest point
    if *path is clear* **then**
        execute the movement
    else
        move to a random point
    end if
end for
return *travel path*

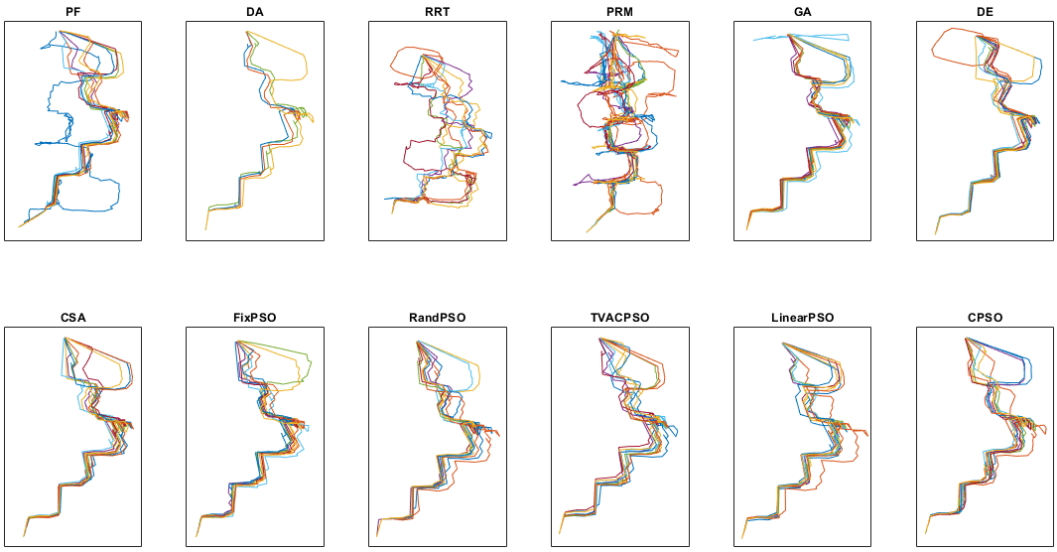


Figure S1: **Trajectory traces of all approaches in Experiment I. Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.**

Figs S1, S2, S3, S4 provides complimentary results to Figs 6, 9, 12, 14 in the main article respectively.

Fig S1 represents the trajectory traces of methods utilized in experiment I (Maze). Each sub-figure in Fig. S1 and subsequent figures of S2, S3, and S4 represents 10 executions of an specific motion planning and navigation technique (i.e., 10 paths travelled by the turtlebot robot). 10 Different colours are used to distinguish these 10 executions. In all these figures, DA performance is considered as the Benchmark.

From sub-figures in Fig S1, RPM, RRT, and PF are shown to have the highest diversity in the path utilized. Higher level of consistency is observed across meta-heuristic methods and their paths.

Fig S2 illustrates the trajectory traces for employed approaches in Sub-Experiment 1 in experiment II. From sub-figures in Fig S2, CPSO is shown to have the minimum deviations in the path across the 10 executions of the algorithm.

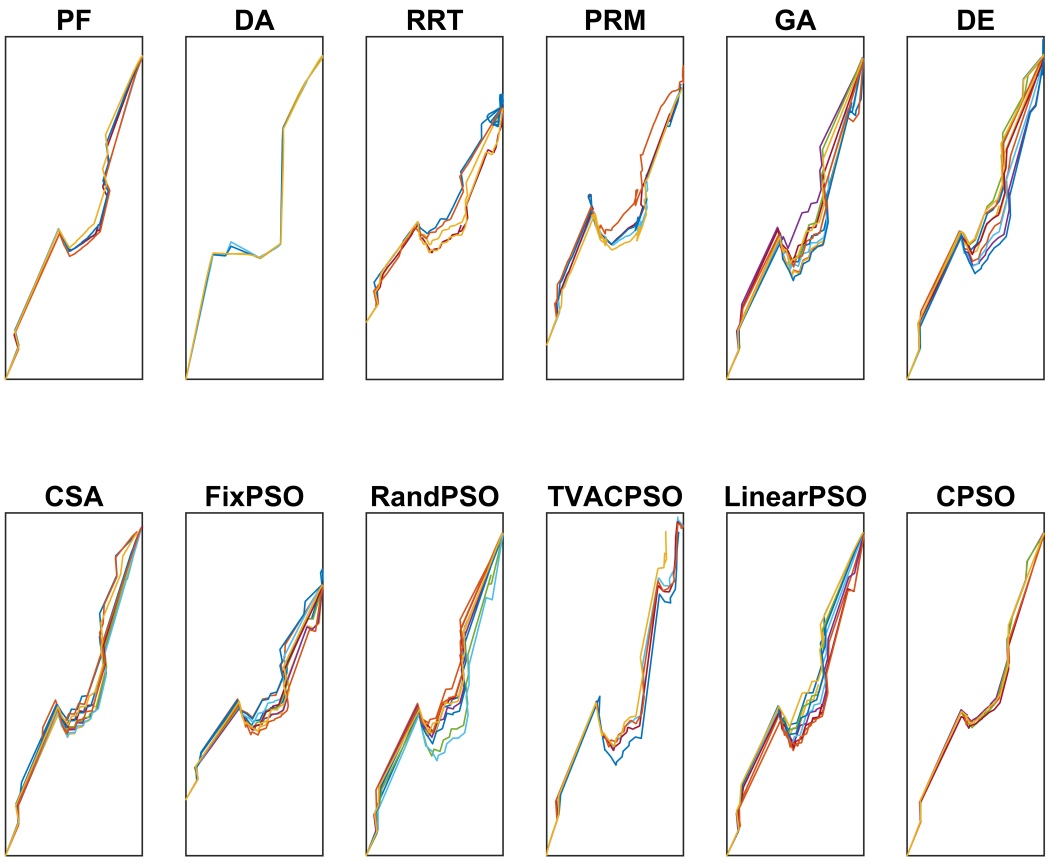


Figure S2: **Trajectory traces of all approaches in Experiment II (Sub-Experiment 1). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.**

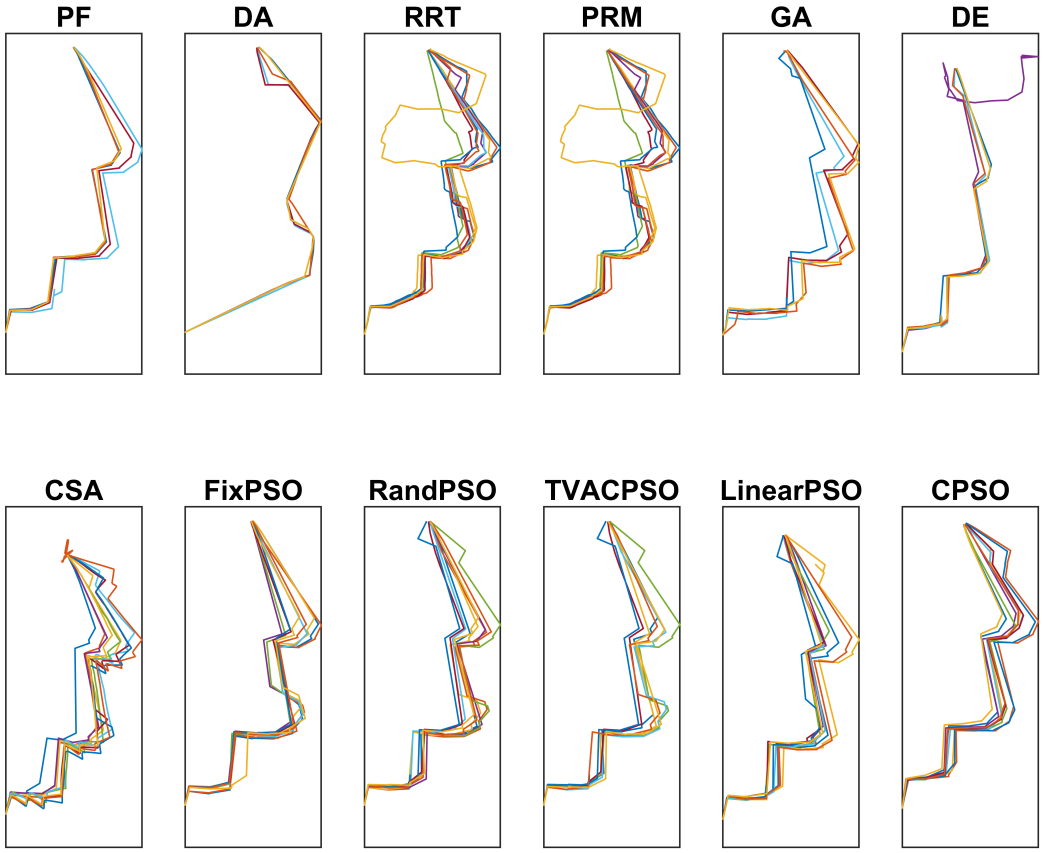


Figure S3: **Trajectory traces of all approaches in Experiment II (Sub-Experiment 2). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.**

Fig S3 illustrates the trajectory traces for employed approaches in Sub-Experiment 2 in experiment II. From sub-figures in Fig S3, RRT, PRM, and in some degree CSA are shown to have the highest level of deviations in the path across the 10 executions of their algorithm. It is noticeable that DE could be considered as the best performing algorithm but due to wandering effect observed in one execution, PF is considered as the most consistent method after the benchmark method DA.

Fig S4 illustrates the trajectory traces for employed approaches in Sub-Experiment 3 in experiment II. In this sub-experiment, CSA demonstrates wandering behavior in late periods of its paths, indicating disability of this method to converge towards the destination.

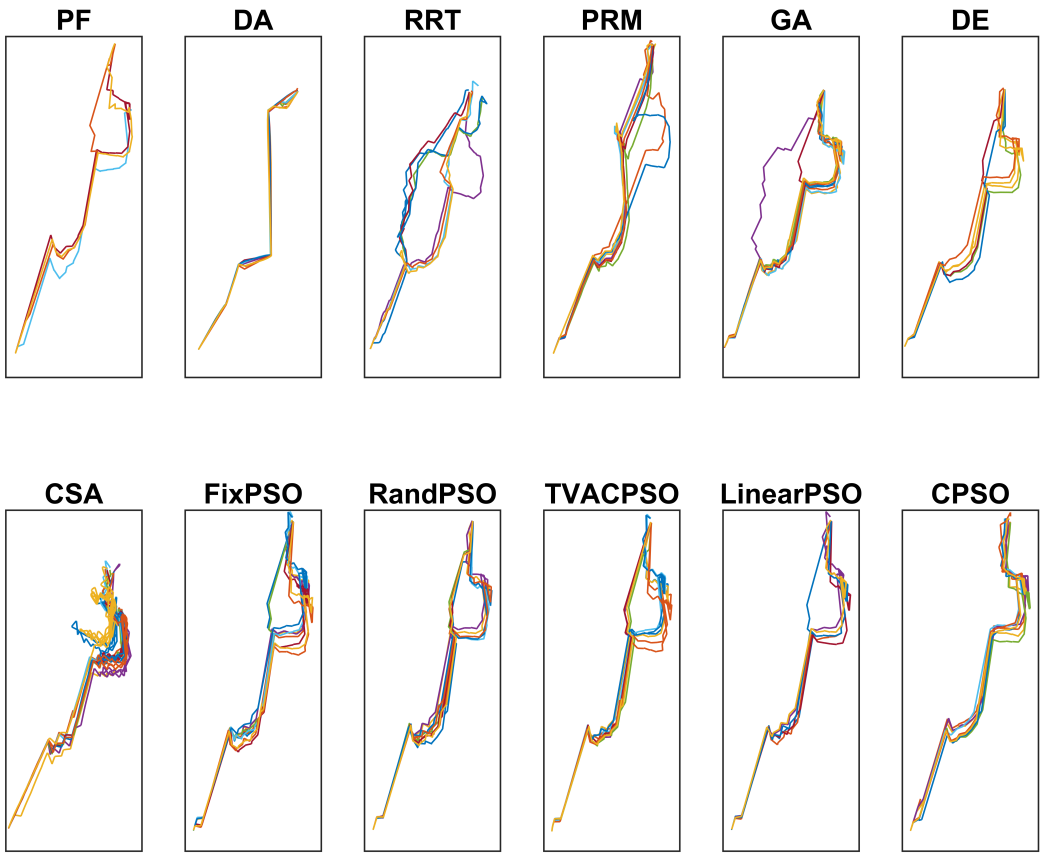


Figure S4: **Trajectory traces of all approaches in Experiment II (Sub-Experiment 3). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.**